Cryptography for Societal Benefit

by

Sunoo Park

Submitted to the Department of Electrical Engineering and Computer Science on May 23, 2018, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Abstract

The deployment of cryptography in society has a range of effects that are not always evident when studying cryptography as a technological construct in isolation. This observation suggests a number of natural research directions that examine cryptography as an instrument of societal influence; that is, as a technological construct in conjunction with its societal effects. This thesis presents the results of six papers spanning the three broad contexts listed next.

- **Institutional accountability**
  Cryptography can enhance transparency and accountability of institutions seeking public trust, such as governmental agencies, judicial systems, and election infrastructure.

- **Individual empowerment in oppressive environments**
  Cryptography can empower individuals to communicate securely and undetectably and to preserve their anonymity, even in hostile environments.

- **Incentivizing collaboration**
  Cryptography can facilitate collaboration between rational — possibly selfish and/or competing — parties in a way that is beneficial to all participants, by providing credible guarantees of secrecy and correct protocol execution to mutually distrustful parties.

Thesis Supervisor: Shafi Goldwasser
Title: RSA Professor of Electrical Engineering and Computer Science
Acknowledgments

First and foremost, I am grateful to my advisor Shafi Goldwasser for being an indescribably open-minded, creative, energetic, and brilliant mentor over the last five years. I have no doubt she will continue to inspire me for years to come. I am grateful to Ron Rivest, Vinod Vaikuntanathan, Ran Canetti, Silvio Micali, and Yael Tauman Kalai for their invaluable collaborations and advice throughout my Ph.D. experience; and I am grateful to Danny Weitzner, Joi Ito, and Jonathan Zittrain for supporting and fueling my more recently developing pursuits in the realm of societal and legal/policy aspects of privacy, security, and technology. I am also grateful to all of my research collaborators, so many of whom have become not only colleagues but also friends — with a special mention to those who coauthored the research presented in this thesis, namely, Pablo Azar, Jonathan Frankle, Shafi Goldwasser, Thibaut Horel, Pavel Hubáček, Silas Richelson, Adam Sealfon, Daniel Shaar, Vinod Vaikuntanathan, and Danny Weitzner — and to my cohort of graduate students in the Theory group at MIT, for their camaraderie and friendship.

I also want to thank Ivan Damgård, Ross Anderson, Moti Yung, and Peter Landrock for sparking and encouraging my interest in cryptography research before graduate school. Looking further back in time, there are many more people who shaped the path that led me to where I am now. I gratefully acknowledge Sean Holden and Arthur Norman’s mentorship during my undergraduate studies; and my most memorable school teachers Roy Jones and Andrew Barker, who nurtured my interest in maths and computer science. Finally, looking back to the very beginning, I am grateful to my parents for their love and support.

Funding acknowledgements. The research in this thesis was supported by the following grants: NSF MACS (CNS-1413920), DARPA IBM (W911NF-15-C-0236), SIMONS Investigator Award Agreement Dated June 5th, 2012, and the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.
Contents

1 Introduction and Preliminaries 13

1 Introduction 15
   1.1 Cryptography for institutional accountability 18
   1.2 Cryptography to empower individuals in hostile environments 20
   1.3 Cryptography to incentivize collaboration 22

2 Preliminaries 25
   2.1 General notation and terminology 26
   2.2 Probability notation for security experiments 28
   2.3 Secret sharing schemes 30
   2.4 Cryptographic commitments 30
      2.4.1 Random oracle as a commitment scheme 33
      2.4.2 Merkle commitment 33
      2.4.3 Pedersen commitment 35
   2.5 Encryption 36
      2.5.1 Public-key encryption 36
      2.5.2 Secret-key encryption 38
   2.6 Zero knowledge 40
      2.6.1 Zero-knowledge SNARKs 41
   2.7 Multi-party computation (MPC) 43
   2.8 Append-only ledgers 46
## II Institutional Accountability

### 3 Practical Accountability of Secret Processes

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>The state of electronic surveillance today</td>
<td>53</td>
</tr>
<tr>
<td>3.2</td>
<td>Related work</td>
<td>58</td>
</tr>
<tr>
<td>3.3</td>
<td>System design</td>
<td>59</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Cryptographic tools</td>
<td>59</td>
</tr>
<tr>
<td>3.3.2</td>
<td>System configuration</td>
<td>61</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Workflow</td>
<td>62</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Additional design choices</td>
<td>64</td>
</tr>
<tr>
<td>3.4</td>
<td>Protocol definition</td>
<td>65</td>
</tr>
<tr>
<td>3.5</td>
<td>Evaluation of MPC implementation</td>
<td>69</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Computing totals in WebMPC</td>
<td>71</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Thresholds and hierarchy with Jiff</td>
<td>72</td>
</tr>
<tr>
<td>3.6</td>
<td>Evaluation of zk-SNARKs</td>
<td>74</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Argument types</td>
<td>75</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Implementation</td>
<td>77</td>
</tr>
<tr>
<td>3.7</td>
<td>Generalization</td>
<td>79</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Generalized framework</td>
<td>81</td>
</tr>
<tr>
<td>3.8</td>
<td>Conclusion</td>
<td>83</td>
</tr>
</tbody>
</table>

### 4 Public Accountability in the Face of Secret Laws

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Secret laws</td>
<td>85</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Our contribution: publicly auditable records on secret data and secret laws</td>
<td>87</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Relation to existing services</td>
<td>94</td>
</tr>
<tr>
<td>4.2</td>
<td>Preliminaries</td>
<td>95</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Blockchains</td>
<td>95</td>
</tr>
<tr>
<td>4.3</td>
<td>Record-keeping scheme</td>
<td>97</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Committing to records</td>
<td>97</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Basic regulatory framework</td>
<td>98</td>
</tr>
</tbody>
</table>
### III Individual Empowerment In Hostile Environments

#### 5 How to Subvert Backdoored Encryption

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Preliminaries</td>
<td>128</td>
</tr>
<tr>
<td>5.1.1 Key exchange</td>
<td>128</td>
</tr>
<tr>
<td>5.1.2 Extractors</td>
<td>131</td>
</tr>
<tr>
<td>5.1.3 Almost pairwise independent hashing</td>
<td>132</td>
</tr>
<tr>
<td>5.2 Subliminal communication</td>
<td>132</td>
</tr>
<tr>
<td>5.3 Impossibility results</td>
<td>139</td>
</tr>
<tr>
<td>5.3.1 Locally decodable subliminal communication schemes</td>
<td>139</td>
</tr>
<tr>
<td>5.3.2 Steganography for adversarial cover distributions</td>
<td>141</td>
</tr>
<tr>
<td>5.4 Construction of the subliminal scheme</td>
<td>142</td>
</tr>
<tr>
<td>5.4.1 Establishing a shared seed</td>
<td>143</td>
</tr>
<tr>
<td>5.4.2 Embedding random strings</td>
<td>148</td>
</tr>
<tr>
<td>5.4.3 Full protocol ((\Phi^<em>, \Xi^</em>))</td>
<td>150</td>
</tr>
<tr>
<td>5.4.4 Proof that ((\Phi^<em>, \Xi^</em>)) is a subliminal communication scheme</td>
<td>153</td>
</tr>
<tr>
<td>5.4.5 On the setup cost and asymptotic rate of ((\Phi^<em>, \Xi^</em>))</td>
<td>154</td>
</tr>
<tr>
<td>5.5 Improving setup cost</td>
<td>156</td>
</tr>
<tr>
<td>5.5.1 Succinct encryption schemes</td>
<td>156</td>
</tr>
<tr>
<td>5.5.2 Next-message distributions with min-entropy</td>
<td>157</td>
</tr>
<tr>
<td>5.6 Steganographic communication schemes</td>
<td>159</td>
</tr>
<tr>
<td>5.6.1 Proof of Theorem 5.3.3</td>
<td>161</td>
</tr>
<tr>
<td>5.7 Open problems</td>
<td>162</td>
</tr>
</tbody>
</table>

#### 6 Repudiability and (Un)claimability of Ring Signatures

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Introduction</td>
<td>165</td>
</tr>
<tr>
<td>6.1.1 Definitional contributions</td>
<td>168</td>
</tr>
<tr>
<td>6.1.2 Overview of our constructions</td>
<td>172</td>
</tr>
</tbody>
</table>
6.1.3 Other related work .......................... 177
6.2 Anonymity and unforgeability of ring signatures .......................... 178
  6.2.1 Anonymity .................................. 179
  6.2.2 Unforgeability ............................... 181
6.3 New definitions: repudiability, claimability, and more .................. 182
  6.3.1 Repudiable ring signatures ...................... 182
  6.3.2 Unrepudiable ring signatures .................. 184
  6.3.3 Claimable ring signatures ...................... 184
  6.3.4 Unclaimable ring signatures ................... 187
  6.3.5 Repudiable-and-claimable ring signatures .............. 188
6.4 Repudiable construction .................................. 191
  6.4.1 Building blocks ................................ 191
  6.4.2 Construction .................................. 194
  6.4.3 Security proof ................................ 196
6.5 Claimable transformation .................................. 210
  6.5.1 Building blocks ................................ 210
  6.5.2 The transformation ................................ 211
6.6 Unclaimable construction .................................. 220
  6.6.1 Lattice trapdoor sampling ............... 220
  6.6.2 Explaining randomness of discrete Gaussian samples ....... 222
  6.6.3 The basic construction of [44] ................. 225
  6.6.4 Unclaimability for the full ring signature scheme of [44] ... 228

IV Incentivizing Collaboration .................................. 230

7 How to Incentivize Data-Driven Collaboration Among Competing Parties 235
  7.1 Overview of contributions ......................... 237
    7.1.1 Discussion and interpretation of our work .......... 241
    7.1.2 Other related work ............................ 244
8.5 Cryptographically blinded games ................................................. 308
  8.5.1 Correspondence of equilibria in blinded games .................. 311
  8.5.2 What can I do with an encrypted action? ......................... 314
8.6 Our Protocols ................................................................. 316
  8.6.1 Cryptographic cheap talk ............................................... 316
  8.6.2 Protocol for computationally bounded players ................. 317
  8.6.3 Protocol for computationally unbounded players ............... 325
  8.6.4 Remarks on efficiency of multi-party computation .......... 329
8.7 Conclusion ........................................................................ 330
Part I

Introduction and Preliminaries
Chapter 1

Introduction

The deployment of cryptography in society has a range of effects that are not always evident when studying cryptography as a technological construct in isolation. This observation suggests a number of natural research directions that examine cryptography as an instrument of societal influence; that is, as a technological construct in conjunction with its societal effects. Such research directions can be meaningful in the context of many other technological constructs, beyond cryptography, and they are complementary (and underrepresented in comparison) to the more well-established computer-science approaches of treating technological constructs primarily as abstract designs to be mathematically analyzed, or as practical systems to be engineered for efficiency and other technical performance metrics.

The significance of the societal consequences of technology have been demonstrated throughout human history, from the invention of the wheel to the Industrial Revolution; and the significance of the societal consequences of computer science have been demonstrated over the last two-thirds of a century, at scale and with a phenomenal pace of change.¹ Widespread access to the Internet, together with other technological developments, has transformed the way that people interact with com-

¹It is remarkable and informative to remember that an IBM mainframe (System/370) computer released in 1970 was priced at approximately a million U.S. dollars [103] and took up an entire room, and this was an example of basically the only type of computer that existed then. (A more recent report suggested that that price when adjusted for inflation would amount to $4–10 million in 2014 [130].)
puting devices\textsuperscript{2} and with other people in their everyday lives, leading to diverse and global consequences as exemplified by the advancement of certain fields of science and medicine resulting from computational studies on large amounts of digital data ("big data studies," e.g., [36, 146, 77]), the role of technology in the Arab Spring [152], the potential of technology to facilitate aid in humanitarian crises [96, 92], and the currently unfolding debate over the impact of propagation of misleading information ("fake news") on social media (e.g., [129]), to name just a few.

Cryptography is intertwined with these technological-societal phenomena. Decades ago, the field’s focus was primarily on a much smaller set of questions than it encompasses today, including how to encrypt information so that only the intended recipient can decrypt it, and how to digitally sign information to endorse it in a provably unforgeable way. These fundamental tools of encryption and digital signatures have come to be used billions of times a day behind the scenes of everyday Internet usage: to encrypt passwords and credit card information as they are sent over the Internet, to encrypt any website content where the browser shows a URL beginning with \texttt{https://}, to verify that some web content indeed comes from the legitimate content provider and not someone else, to verify that smartphone apps and software updates are endorsed by the relevant app store or phone manufacturer, and more.\textsuperscript{3} Modern cryptography, moreover, encompasses the study of a far broader range of more techniques for securing information more generally and to meet more complex requirements. The work in this thesis builds on several examples of modern cryptographic tools, but what is represented herein constitute only a sampling of the field at large. To me, cryptography is fascinating because it “rearranges power: it configures who can do what, from what” [156],\textsuperscript{4} and because of its potential to reconcile

\textsuperscript{2} Including computers, smartphones, smart watches, and any other “smart” devices that perform computing such as televisions, smart home devices, and whatever else the future may hold.

\textsuperscript{3} These examples may serve to illustrate that the nature and scope of cryptography are strikingly different from certain popular misperceptions of cryptography as a shady toolset that is primarily used for purposes along the lines of “subverting the system” and most commonly used by criminal enterprises or terrorists.

\textsuperscript{4} The quote is from Phillip Rogaway’s 2015 essay titled “The Moral Character of Cryptographic Work,” in which he continues as follows: “This makes cryptography an inherently political tool, and it confers on the field an intrinsically moral dimension.”
seemingly conflicting and paradoxical goals: for instance, how to keep information secret while simultaneously making convincing claims about its contents, or how to distribute information across multiple stakeholders such that it is irrecoverable unless many of them collaborate to recover it.

This thesis comprises a collection of research papers studying the design of systems that use cryptography specifically towards achieving desirable societal outcomes or demonstrating a societally significant point. The thesis is organized into three parts that approach this design space based on the interests of different societal actors: institutions (such as the judicial system or election system), individuals, and groups of potential collaborators encompassing institutions, businesses, and/or individuals each with their own self-interested goals. The first part examines how to combine existing cryptographic tools to build systems that serve the goals of institutional accountability and transparency, focusing specifically on the transparency needs of two U.S. surveillance laws\(^5\) about which accountability concerns have been expressed by individuals within the court system itself. The second part considers questions of a more theoretical nature, proposing new cryptographic definitions and constructions thereof, broadly motivated by the two separate concerns of: (1) individual freedom to communicate in secrecy, and (2) how to combine anonymity with credibility. The third and last part draws upon ideas from game theory as well as cryptography, to model the individuals' rational interests and achieve beneficial outcomes for all parties under said models.

The problems studied and solutions proposed herein are intended to be largely decoupled from the policy questions of how a given system should operate as a matter of principle, in the following sense. Naturally, when I devote research efforts to projects about practical cryptographic approaches to adding accountability and transparency to surveillance systems, the work is accompanied by a belief that accountability and transparency are worthwhile and interesting goals. At the same time, the work does not take a position about whether the underlying surveillance should be happening

\(^5\)The Electronic Communications Privacy Act (ECPA) [69] and the Foreign Intelligence Surveillance Act (FISA) [74].
and in what form; these are certainly important questions, but outside the scope of the work, whose intent is to ask in a way agnostic to the aforementioned concerns: given a system that may already be in place, how could it be augmented using cryptography to further the goals of accountability and transparency? This is just one approach among many possible and complementary approaches to the study of cryptography in the context of its societal effects.

Each of the three parts of the thesis comprises two papers, each presenting a cryptographic solution to a specific problem motivated by the interests of institutions, individuals, or potentially collaborative groups within a society. The rest of the Introduction gives an overview of the results presented under each theme and chapter. Full constructions and proofs are given within the chapters, as well as more detailed informal descriptions of the results.

1.1 Cryptography for institutional accountability

Accountability and transparency are often a desirable goal, in principle, both for institutions and the people they serve. However, institutions often deal with information that is believed best kept secret for a variety of reasons, including national security, privacy, legal considerations, or democratic values (e.g., the secret ballot). Traditionally, such secrecy has necessarily diminished the transparency and public accountability possible with respect to the secret data, in many cases to essentially zero. It is reasonable to question, for example: how believable is a spoken assertion that some unseen classified data adheres to all required regulations, especially when it comes from an institutional official to a large population with whom he or she is not personally acquainted?

Modern cryptography transforms the space of possibilities. Most importantly, it overturns the longstanding assumption that assertions about secret data cannot be made in a credible fashion. This assumption has long been a tenet of the very design of institutional systems, typically in an implicit fashion.

This thesis presents two systems designed to promote accountability in institutions
by using cryptography to enable credible assertions about secret data, inspired by specific use cases in the U.S. legal system, which is currently grappling with issues of accountability and transparency. For example, the presiding judge of the FISA Court, Judge Reggie B. Walton, expressed concern in 2013 that “the court lacks the tools to independently verify how often the government’s surveillance breaks the court’s rules that aim to protect Americans’ privacy” [121].

The two systems presented in this thesis are designed to respond to the following question: supposing that institutions are benevolent and desire accountability and transparency, how can cryptography help them achieve these goals? The first comes from a paper titled “Practical Accountability of Secret Processes” that I coauthored with Jonathan Frankle, Daniel Shaar, Shafi Goldwasser, and Daniel J. Weitzner [78], and the second comes from a paper titled “Public Accountability vs. Secret Laws: Can They Coexist?” that I coauthored with Shafi Goldwasser [89].

The first system [78] puts forth a general framework to release limited information subject to certain accountability goals about the data passing through a generic secret information process. This framework is presented by way of the concrete and timely example of accountability in electronic surveillance authorized by the U.S. Federal Court System, which is currently discussing strategies to improve the accountability of that very process: specifically, we design and implement a prototype of a system to increase transparency and accountability for one of the leading United States electronic surveillance laws, the Electronic Communications Privacy Act (ECPA) [69]. Our framework can, moreover, be generalized to a much broader class of institutional processes involving secret information.

The second system [89] follows a similar framework, but provides the additional capability not only making assertions about publicly stated properties of secret data, but also credible claims about secret properties of secret data, tailored to the Kafkaesque setting of “secret laws” — when even regulations themselves that govern the secret data are considered to be sensitive information. Our consideration of this scenario is motivated by the documented phenomenon of classified rulings under the Foreign Intelligence Surveillance Act (FISA) [74] by the U.S. Foreign Intelligence Surveillance
Court. In contrast to [78], [89] is a more theoretical proposal without an implementation, and also proposes additional optional features such as automatic imposition of penalties when the accountability systems reveals that some (possibly secret) regulations were not satisfied by some institutional action.

1.2 Cryptography to empower individuals in hostile environments

The next part considers the potential of cryptography to empower individuals to take control of their privacy and anonymity in the face of adverse conditions involving potentially malicious institutions. This strikes a contrast with the previous part, which considered how cryptography can help institutions in the case that they support the benevolent goals of accountability and transparency.

The first chapter in this part presents a paper titled “How to Subvert Backdoored Encryption” that I coauthored with Thibaut Horel, Silas Richelson, and Vinod Vaikuntanathan. We prove a possibility result in an Orwellian hypothetical model, showing that any attempt at governmental surveillance of citizens’ communications would require either constant surveillance of all devices (not just the messages sent between them) or settling for unencrypted communication that would be open to spying by foreign powers. Slightly more precisely, we show a way for individuals to communicate securely and undetected by the government without any setup assumptions, provided that the government (1) permits communication using a semantically secure encryption scheme (a reasonable supposition if they do not want to open all domestic communications to foreign spying) and (2) is not constantly surveilling end-point devices. This result holds even if the government can choose the encryption scheme (after seeing the description of the subliminal communication scheme) and has a copy of every decryption key for encryption scheme. The extreme nature of the hypothetical surveillance state we consider is not because such a situation is necessarily realistic, but because a possibility result that holds even in such a dire scenario
is a stronger statement than one which is designed against weaker adversaries.

The possibility of covert communication in such adverse circumstances is an observation of timely interest in the context of the ongoing policy debate about governmental access to encrypted information: certainly, surveilling every computer and device would incur vastly more resources and present more practical challenges than “merely” surveilling all communication channels. The practical and policy implications of our theoretical finding, however, should be considered in light of the fact that governmental interest is likely much more strongly in surveilling most communications and/or making evading surveillance onerous, rather than surveilling every single communication that occurs.

Next, we shift our attention from encryption to signatures. Ring signatures [154], introduced by Rivest, Shamir, and Tauman Kalai, are a variant of digital signatures which certify that one among a particular set of parties has signed a message while hiding which party in the was the signer. They can be useful, for example, to certify that some leaked information comes from a privileged set of government or company officials without revealing who the whistleblower is — an important capability in the face of potentially adversarial institutions — or to issue important orders or directives without setting up the signer to be made a scapegoat for repercussions. Ring signatures are deliberately designed to allow anyone to attach anyone else’s name to a signature, without the latter’s knowledge or consent.

The chapter presents a paper titled “It Wasn’t Me! Repudiability and (Un)claimability of Ring Signatures” that I coauthored with Adam Sealfon. It asks the following question: what guarantee does a ring signature scheme provide if a purported signatory wishes to denounce a signed message—or alternatively, if a signatory wishes to later come forward and claim ownership of a signature? Given the motivation of anonymity behind the notion of a ring signature, a natural first intuition might be that parties should be able neither to denounce or claim a signature in a convincing way. That way, even if an adversary were able to place all members of a ring under duress, he could not obtain convincing evidence of the real signer’s identity.

However, prior security definitions for ring signatures do not give a conclusive
answer to this question. That is, a non-signer might be able to repudiate a signature that he did not produce, or this might be impossible. Similarly, a signer might be able to later convincingly claim that a signature he produced is indeed his own, or not. While the inability to repudiate or claim signatures is a useful guarantee, as discussed above, it is not the only reasonable guarantee imaginable, as we discuss next. In any case, however, a guarantee in one direction or the other seems more desirable than no guarantee either way.

In fact, it can be argued that any of these guarantees might be desirable: for instance, a whistleblower might have reason to want to later claim an anonymously released signature; on the other hand, it might be desirable that even under duress, a member of a ring cannot produce proof that he did or did not sign a particular signature. We formalize definitions and give constructions of the new notions of repudiable, claimable, and unclaimable signatures. (The fourth notion of unrepudiability is implied by unclaimability.)

1.3 Cryptography to incentivize collaboration

The final part of the thesis focuses on the use of cryptography to enable and incentivize collaboration by self-interested and possibly competing parties, in order to achieve beneficial outcomes for both individuals and larger groups of parties.

The first chapter in this part presents a paper titled “How to Incentivize Data-Driven Collaboration Among Competing Parties” that I coauthored with Pablo Azar and Shafi Goldwasser [11]. The work focuses on a stylized model of data-driven collaboration between competing parties each seeking to get a guaranteed share of “credit” for any finding that results from the collaboration. We say that a mechanism for collaboration achieves collaborative equilibrium if it guarantees a higher reward for each participant when collaborating than when working alone: a perhaps seemingly paradoxical outcome, whose study we motivate by the recent phenomenon of so-called “big data” studies (e.g., machine learning on genome data) yielding results far more informative than seems possible with many smaller datasets analyzed separately.
We show that computing collaborative equilibria is in general NP-complete, but also give mechanisms compute collaborative equilibria for a range of natural model settings. Based on potential collaborators’ private inputs, our mechanisms first determine whether a collaborative equilibrium is feasible, then issue outputs to participants in a specific ordering, with delays in between outputs. The mechanism’s correct behavior and the enforcement of delays are ensured by secure multi-party computation and a generalization of time-lock puzzles which we call time-line puzzles. The original notion of time-lock puzzles was proposed by May [133] then formalized and constructed by Rivest, Shamir, and Wagner [155].

The next and final paper we present is titled “Cryptographically Blinded Games: Leveraging Players’ Limitations for Equilibria and Profit,” and coauthored with Pavel Hubáček. It proposes a way to implement coarse correlated equilibria (CCE) using multi-party computation and non-malleable encryption. CCE represents a class of outcomes of interactions between self-interested parties, which can yield higher payoffs for everyone involved, compared to the older and better-known notions of Nash or correlated equilibria. However, the traditional formulation of CCE requires a mediator to coordinate the parties’ actions, and it may realistically be costly or impractical to procure a mediator in whose impartiality all parties trust. Our protocol eliminates the need for such a mediator.

The essence of our approach is to exploit the power of encryption to selectively restrict the information available to players about sampled action profiles, such that these desirable equilibria can be stably achieved. In contrast to previous applications of cryptography to game theory, this work is the first to employ the paradigm of using encryption to allow players to benefit from hiding information from themselves, rather than from others.
Chapter 2

Preliminaries

This chapter establishes general notation and terminology, and introduces syntax and security definitions for cryptographic primitives that are used throughout the remainder of the thesis.

Organization. Section 2.1 gives general notation and terminology; Section 2.2 introduces the probability notation used in this thesis to define security experiments (and also defines one-way functions); Section 2.3 presents secret-sharing schemes; Section 2.4 discusses cryptographic commitments; Section 2.5 presents encryption; Section 2.6 introduces succinct zero-knowledge arguments; Section 2.7 defines secure multi-party computation; and Section 2.8 introduces public ledgers.

In most cases where a particular definition or primitive is used only within the context of a single chapter, it is introduced in a preliminaries section within that chapter, rather than in this section. For the sake of completeness, I list here the primitives that are introduced in later chapters: Section 4.2 presents blockchains as a specific type of public ledger; Section 5.1 defines key exchange, randomness extractors, and almost pairwise independent hashing; Sections 6.4.1, 6.5.1, 6.6.1, and 6.6.2 define verifiable random functions (VRFs) and ZAPs (which are a type of two-message proof system), signatures and pseudorandom functions, and certain lattice assumptions and algorithms; Section 7.4.3 defines time-lock puzzles; Section 8.2 introduces standard game-theoretic notions of extensive games and equilibria; and Section 8.3 gives perfect
and computational definitions of non-malleable encryption.

Most of the chapters additionally introduce new definitions that are novel to the work being presented therein, and thus not considered to be preliminaries.

2.1 General notation and terminology

The thesis uses the following textual abbreviations: “s.t.” for “such that,” “w.l.o.g.” for “without loss of generality,” “i.i.d.” for “independent and identically distributed,” “w.p.” for “with probability,” and “resp.” for “respectively.”

Sets. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$, and $[n] \rightarrow [n]$ denotes the set of all permutations of $[n]$. The symbol $\sqcup$ denotes the disjoint union operation. For a set $S$, $\mathcal{P}(S)$ denotes the powerset of $S$ and $\Delta(S)$ denotes the set of all distributions over $S$.

Distributions and random variables. For a finite set $S$, we write $s \leftarrow S$ to denote that $s$ is drawn uniformly at random from $S$. $\mathcal{U}(S)$ denotes the uniform distribution over $S$.

For $n \in \mathbb{N}$, $\mathcal{U}_n$ is a uniform variable over $\{0, 1\}^n$. We write $\mathcal{X} \sim \mathcal{Y}$ to express that $\mathcal{X}$ and $\mathcal{Y}$ are identically distributed. Given two variables $\mathcal{X}$ and $\mathcal{Y}$ over $\{0, 1\}^n$, we denote by $\|\mathcal{X} - \mathcal{Y}\|_s$ the statistical distance defined by:

$$
\|\mathcal{X} - \mathcal{Y}\|_s = \frac{1}{2} \sum_{x \in \{0, 1\}^n} |\Pr[\mathcal{X} = x] - \Pr[\mathcal{Y} = x]| = \max_{S \subseteq \{0, 1\}^n} \|\Pr[\mathcal{X} \in S] - \Pr[\mathcal{Y} \in S]\|.
$$

For a random variable $\mathcal{X}$, $\mathbb{E}[\mathcal{X}]$ denotes its expectation. We define the min-entropy of $\mathcal{X}$ by

$$
H_\infty(\mathcal{X}) = -\log \max_x \Pr[\mathcal{X} = x].
$$

The collision probability is $\mathbb{C}(\mathcal{X}) \triangleq \sum_x \Pr[\mathcal{X} = x]^2$.

For an event $E$, $1[E]$ denotes the indicator variable of $E$.

The relation $\approx_c$ denotes computational indistinguishability, and the relation $\approx_s$ denotes statistical indistinguishability.
**Algorithms and asymptotics.** For a randomized algorithm \(A\), when we refer to the algorithm’s randomness explicitly, we write \(A(x; \rho)\) to denote the output of \(A\) on input \(x\) and randomness \(\rho\). Unless otherwise stated, we assume that the domain of randomness is \(\{0, 1\}^\ell\) for some \(\ell \in \mathbb{N}\). We write \(\$A\) to denote the length of randomness that \(A\) takes: e.g., in the preceding example, \(\$A = \ell\).

We write “PPT” to denote probabilistic polynomial time, and we call distributions that can be sampled in probabilistic polynomial time “PPT-samplable”. We sometimes refer to PPT algorithms as *efficient* algorithms. Cryptography against computationally bounded adversaries traditionally deals with algorithms that are PPT in a *security parameter*. \(\lambda\) denotes the security parameter throughout, and PPT algorithms may be thought to receive as input the security parameter in unary, \(1^\lambda\). The security parameter is the only variable in the thesis with global scope; all other variables are scoped within chapters, although the chapters use similar notation and variable naming conventions for readability’s sake. This input \(1^\lambda\) is often omitted and left implicit.

\(\text{poly}(n)\) denotes the set of polynomials in \(n\), and \(\text{negl}(n)\) denotes the set of negligible functions in \(n\). respectively, polynomial or negligible. A negligible function is one that is asymptotically smaller than all inverse polynomials, as defined formally below.

**Definition 2.1.1 (Negligible).** A function \(\varepsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}\) is negligible if

\[
\forall c \in \mathbb{N}, \exists \Lambda \in \mathbb{N} \text{ s.t. } (\lambda > \Lambda \Rightarrow \varepsilon(\lambda) < \lambda^{-c})
\]

Where the argument of \(\text{negl}\) or \(\text{poly}\) is omitted, it may be assumed to be the security parameter. Other functions of the security parameter may be similarly abbreviated: e.g., \(n\) instead of \(n(\lambda)\). Such abbreviations are sometimes made explicit by writing \(n = n(\lambda)\) when introducing \(n\).

**Miscellaneous.** The operation \(\oplus\) denotes (bitwise) exclusive-or. The operations \(\land\) and \(\lor\) denote conjunction and disjunction respectively. \(\circ\) denotes function compo-
position, and for a function $f$, we write $f^t$ to denote the $t$-fold composition of $f$, i.e., $f \circ f \circ \cdots \circ f$.

Logarithms are base 2 unless otherwise indicated. The binary entropy function $H_{\text{bin}} : [0, 1] \rightarrow \mathbb{R}$ is defined by

$$H_{\text{bin}}(p) = \begin{cases} -p \log(p) - (1 - p) \log(1 - p) & \text{if } p \notin \{0, 1\} \\ 0 & \text{if } p \in \{0, 1\} \end{cases}.$$

The symbol $s$ is used to denote state that is passed between different parts of a stateful algorithm. The symbol $\bot$ is used to denote a “null” value that is not contained in any other sets unless explicitly stated.

### 2.2 Probability notation for security experiments

Security definitions in cryptography are often described in terms of a “game” played between a challenger and an adversary, in which the adversary is considered to “win” if it can violate the desired security properties of the primitive in question. A security definition typically stipulates an upper bound on the adversary’s probability of “winning.” The game may be referred to as a security game or security experiment.

The following syntax is used to denote the probability that certain conditions (on the right-hand side of the colon) are satisfied by random variables following a particular distribution (on the left-hand side). The distribution on the left may be thought to describe the security game, and the conditions on the right describe the event whose probability is of interest (e.g., the adversary’s “winning” event).

$$\Pr \left[ \text{distributions of random variables : conditions on random variables} \right]. \quad (2.1)$$

This notation is exemplified in the definition of a one-way function, below.

**Definition 2.2.1** (One-way function). A function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is one-way if the following two conditions hold.
1. \textbf{(Easy to compute)} There is a PPT algorithm \( M \) s.t. for all \( x \), \( M(x) = f(x) \).

2. \textbf{(Hard to invert)} For all PPT algorithms \( A \), there is a negligible function \( \varepsilon \) s.t. for all \( \lambda \in \mathbb{N} \),

\[
\Pr \left[ \begin{array}{c}
x \leftarrow \{0,1\}^\lambda \\
x' \leftarrow A(1^\lambda, f(x))
\end{array} : f(x) = f(x') \right] \leq \varepsilon(\lambda) .
\]

\[(2.2)\]

\textbf{Example 2.2.2.} The "hard to invert" condition of Definition 2.2.1 could equivalently be expressed in terms of a security game between a challenger and adversary.

\[\begin{array}{c}
\textbf{The one-way function experiment } \text{Expt}^\text{OWF}_\lambda \\
1. \text{The challenger samples a random } x \leftarrow \{0,1\}^\lambda \text{ and sends } f(x) \text{ to the adversary.} \\
2. \text{The adversary outputs some } x' \in \{0,1\}^\lambda. \\
3. \text{The output of the experiment is } 1 \text{ if } f(x) = f(x'), \text{ and } 0 \text{ otherwise.}
\end{array}\]

Using this notation, \[(2.2)\] could equivalently be expressed as:

\[
\Pr[\text{Expt}^\text{OWF}_\lambda = 1] \leq \varepsilon(\lambda) .
\]

The rest of the thesis uses primarily the notation of \[(2.1)\] rather than defining security experiments explicitly as in Example 2.2.2.

\textbf{Remark 2.2.3} (Implications of one-way functions). The works in this thesis rely on the existence of one-way functions as the underlying assumption for the security of various constructions. The relevant implications for the purpose of this thesis are that one-way functions imply pseudorandom generators, pseudorandom functions, commitments, and reusable secret-key encryption (i.e., secret-key encryption where a key can be reused for many encryptions, unlike a one-time pad) \cite{39, 86, 87, 97}.
2.3 Secret sharing schemes

A secret sharing scheme specifies a method for a designated party (the “dealer”) to share a secret $s$ among $N$ players so that only large enough subsets of players can reconstruct the secret value $s$. The dealer gives privately a share $s_i$ to each player $i$, so that any set of up to $k - 1$ shares contains no information about $s$; however, it can efficiently be reconstructed given any $k$ or more shares. The formal definition follows.

**Definition 2.3.1** (Secret sharing scheme [163]). A $k$-out-of-$N$ secret sharing scheme is a pair of algorithms $(\text{Share}, \text{Recon})$ as follows. Share takes as input a secret value $s$ and outputs a set of shares $S = \{s_1, \ldots, s_N\}$ such that the following properties hold.

- *(Correctness)* For any $S' \subseteq S$ s.t. $|S'| \geq k$, it holds that $\text{Recon}(S') = s$.
- *(Secrecy)* For any $S' \subseteq S$ s.t. $|S'| < k$, it holds that $H_{\text{bin}}(s) = H_{\text{bin}}(s|S')$.

Recon takes as input a (sub)set $S'$ of shares and outputs:

$$\text{Recon}(S') = \begin{cases} \bot & \text{if } |S'| < k \\ s & \text{if } \exists S \text{ s.t. } S' \subseteq S \text{ and } S \in \text{Share}(s) \text{ and } |S'| \geq k \end{cases}.$$  

**Remark 2.3.2.** Efficient secret-sharing schemes for any $N$ and $k < N$ exist unconditionally [163].

2.4 Cryptographic commitments

A cryptographic commitment $c$ is a string generated from some input data $D$, which has the properties of hiding and binding. Informally, $c$ by itself must reveal no information about the value of $D$, but also, $D$ must be able to be revealed or “opened” (by the person who created the commitment) in such a way that any observer can be sure that $D$ is the data with respect to which the commitment was made. We refer to $D$ as the content of $c$.

Throughout this thesis, we use the term “commitment scheme” to refer only to non-interactive commitment schemes. A formal definition follows.
Definition 2.4.1. A commitment scheme is a triple of PPT algorithms \( C = (\text{Setup}, \text{Commit}, \text{Open}) \) with the following syntax.

- **Setup**\((1^\lambda)\) takes as input a security parameter \( \lambda \) (in unary) and outputs public parameters \( pp \).
- **Commit**\((pp, m; \rho)\) takes as input \( pp \), a message \( m \), and randomness \( \rho \). It outputs a commitment \( c \).
- **Open**\((pp, m', c, \rho')\) takes as input \( pp \), a message \( m' \), and commitment randomness \( \rho' \). It outputs 1 if \( c = \text{Commit}(pp, m', \rho') \) and 0 otherwise.

A secure commitment scheme is required to satisfy the following properties.

- **(Hiding)** For all PPT adversaries \( A = (A_1, A_2) \), \( \exists \) negligible \( \varepsilon \) s.t. \( \forall \lambda \in \mathbb{N} \),
  \[
  \Pr \left[ \begin{array}{c}
  pp \leftarrow \text{Setup}(1^\lambda) \\
  (m_0, m_1, s) \leftarrow A_1(1^\lambda) \\
  b \leftarrow \{0, 1\} \\
  c \leftarrow \text{Commit}(pp, m_b) \\
  b' \leftarrow A_2(c, s)
  \end{array} \right] : b' = b \leq 1/2 + \varepsilon(\lambda). \quad (2.3)
  \]

If \( C \) moreover satisfies (2.3) for computationally unbounded adversaries and \( \varepsilon = 0 \), then \( C \) is said to satisfy perfect hiding.

- **(Binding)** For all PPT adversaries \( A \), \( \exists \) negligible \( \varepsilon \) s.t. \( \forall \lambda \in \mathbb{N} \),
  \[
  \Pr \left[ \begin{array}{c}
  pp \leftarrow \text{Setup}(1^\lambda) \\
  (c, m, \rho, m', \rho') \leftarrow A(1^\lambda) \\
  b \leftarrow \text{Open}(pp, (c, m, \rho)) \\
  b' \leftarrow \text{Open}(pp, (c, m', \rho'))
  \end{array} \right] : m \neq m' \land b = 1 = b' \leq \varepsilon(\lambda). \quad (2.4)
  \]

If \( C \) moreover satisfies (2.4) for computationally unbounded adversaries and \( \varepsilon = 0 \), then \( C \) is said to satisfy perfect binding.

For our purposes, the public parameters \( pp \) are assumed to be generated in an initial
setup phase and thereafter publicly known to all parties, so we sometimes leave them implicit and write simply Commit(m) and Open(c, m, ω) for brevity.

**Remark 2.4.2.** Some commitment schemes do not have a Setup algorithm and consist simply of a pair of algorithms (Commit, Open). In this case, the hiding and binding properties of Definition 2.4.1 should be parsed as if the public parameters pp are empty. The Setup algorithm is included in Definition 2.4.1 as the thesis later makes reference to some commitment schemes which do have a Setup algorithm.

**Remark 2.4.3.** For completeness, it is worth noting that there is also an intermediate notion of statistical hiding and binding, which are stronger than the respective computational variants but weaker than the respective perfect variants. Formal definitions of the statistical versions are omitted as they are unnecessary for the thesis.

It is well known that a commitment scheme cannot satisfy both statistical hiding and statistical binding simultaneously. Therefore, it is also impossible to simultaneously achieve perfect hiding and binding; that is, at least one of the guarantees must be computational.

**Remark 2.4.4.** Definition 2.4.1 assumes that the decommitment information ρ' which is input to the Open algorithm is equal to the commitment randomness. This assumption is without loss of generality in the following sense: given the binding property, the commitment randomness certainly suffices to convince a verifier that a particular message is indeed the one that the committer committed to. In general, however, the decommitment information or opening does not necessarily consist of the commitment randomness it may be desirable for efficiency or other reasons to construct commitment schemes whose decommitment information has a different form. The above definition suffices for this thesis and is presented thus to simplify notation.

Note that since we have defined the behavior of Open already above, in concrete definitions of commitment schemes in the rest of the thesis, it suffices to provide descriptions just of the Setup and Commit algorithms.

**Definition 2.4.5** (Succinctness of commitment). A commitment scheme C is succinct if the size of commitments is independent of the message size.
2.4.1 Random oracle as a commitment scheme

A random oracle is a truly random function from a domain to a range. In the random oracle model, all algorithms are assumed to have access to the random oracle, i.e., to an oracle that on each input outputs the value of the random oracle evaluated at that input. The random oracle model gives rise to a simple computationally hiding and binding and succinct commitment scheme. In practice, if one is willing to assume that a concrete hash function such as SHA-2 behaves like a random oracle, the hash function can serve as a very efficient and succinct commitment scheme.

Instead of running the hash function directly on the data $D$ to be committed, one should use $D||\rho$ as the input to the hash function, where $\rho \leftarrow \{0, 1\}^\lambda$ serves as commitment randomness. Essentially, the use of $\rho$ ensures that two commitments to the same $D$ will be indistinguishable from two commitments to different values.

2.4.2 Merkle commitment

This subsection describes a succinct type of commitment called Merkle commitment. The definition of a collision-resistant hash function family is presented first, as a prerequisite.

**Definition 2.4.6** (Collision-resistant hash function family). A hash function family $\mathcal{H} = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ consists of functions $H_\lambda : \{0, 1\}^{\ell_{\text{seed}}(\lambda)} \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^{\ell_{\text{out}}(\lambda)}$ which map a $\ell_{\text{seed}}(\lambda)$-bit seed and a $\lambda$-bit input to a $\ell_{\text{out}}(\lambda)$-bit output. We write $h_{\lambda,s}$ to denote the partial function $H_\lambda(s, \cdot)$.

$\mathcal{H}$ is collision-resistant if for all PPT adversaries $A$, there is a negligible function $\varepsilon$ s.t. for all $\lambda \in \mathbb{N}$,

$$\Pr \left[ s \leftarrow \{0, 1\}^{\ell_{\text{seed}}(\lambda)} : (x, x') \leftarrow A(1^\lambda, s) \land x \neq x' \land h_s(x) = h_s(x') \right] \leq \varepsilon(\lambda).$$

1To be precise, $\rho \leftarrow \{0, 1\}^{\Omega(\log(\lambda))}$ suffices.
2In this definition, for simplicity, we assume the seed is to be sampled uniformly from $\{0, 1\}^{\ell_{\text{seed}}(\lambda)}$. The most general definition of collision-resistant hash function allows for arbitrary PPT-samplable seed distributions; the curious reader may refer to [112] for an example of such a definition.
We say $\mathcal{H}$ is moreover length-halving if $\ell_{\text{out}}(\lambda) \leq \lambda/2$.

**Definition 2.4.7 (Merkle commitment).** Given any commitment scheme

$$C = (\text{Setup, Commit, Open})$$

and a length-halving collision-resistant hash function family $\mathcal{H} = \{H_{\lambda}\}_{\lambda \in \mathbb{N}}$, the Merkle commitment $C_{\text{Merkle}}$ is defined as follows.

- **Setup**$_{\text{Merkle}}(1^\lambda)$ outputs $(pp, h)$ where $pp \leftarrow \text{Setup}(1^\lambda)$ and $h(\cdot) = H_{\lambda}(s, \cdot)$ for a random seed $s$.
- **Commit**$_{\text{Merkle}}((pp, h), m)$ divides the message $m$ into $B/2$-bit contiguous chunks, $m_1, \ldots, m_{\lceil |m|/B \rceil}$, computes $c_i = \text{Commit}(pp, m_i; \omega_i)$ for random $\omega_i$, and considers the commitments $c_1, \ldots, c_{\lceil |m|/B \rceil}$ to be "labels" on the leaves of a binary tree. The label of any node $\nu$ in the binary tree is then defined to be $\ell_{\nu} = h(\ell_{\nu,0} || \ell_{\nu,1})$ and $\ell_{\nu,0}, \ell_{\nu,1}$ are defined to be the labels of $\nu$'s children. Finally, the output is $\ell_{\text{root}}$ where $\text{root}$ denotes the root node of the tree.

**Remark 2.4.8.** In the case of Merkle commitment, not all of the commitment randomness needs to be passed as input to the opening algorithm; rather, it would suffice for the opening algorithm to take as input the set of all labels in the tree computed by $\text{Commit}_{\text{Merkle}}$ — and check the equality of the root node label with the commitment, and also check that each node's claimed label is indeed the hash of its two children's labels. In practice, this is the preferable implementation (see Remark 2.4.4 for more discussion).

**Partial opening.** It is possible to open any consecutive pair $(m_{2i-1}, m_{2i})$ of the $B/2$-bit chunks of the message $m$ by revealing the labels of the nodes along its path to the root (together with the labels of immediate siblings of those nodes). This opening procedure provably preserves the hiding property for all chunks of the message apart from $m_{2i-1}$ and $m_{2i}$. 
Generalized Merkle commitment. In Definition 2.4.7, the chunk sizes are fixed at $B/2$, but this scheme would work equally well with arbitrary, variable-size chunks. That is, the data could be split into chunks according to logical divisions, and the minimum appropriate number of chunks can be decided adaptively, case by case. (Reducing the number of chunks is beneficial for efficiency.)

Random-oracle-based Merkle commitment. To use a random oracle to instantiate Merkle commitment, the randomized commitment scheme described in Section 2.4.1 above should be used to generate the commitments for leaf nodes, and the random oracle can be applied directly (i.e., without any "commitment randomness") in place of $h$ at each non-leaf node of the tree. This is because a random oracle is a collision-resistant hash function.

Perfectly hiding Merkle commitment. If a perfectly hiding commitment scheme (such as the Pedersen commitment described in Section 2.4.3) is used to commit leaf node data, then the resulting commitment scheme is both succinct and perfectly hiding. This is regardless of what hash function family is used (and can be done in conjunction with the random-oracle-based hashing described in the previous paragraph).

2.4.3 Pedersen commitment

Possibly the best-known perfectly hiding commitment scheme is the Pedersen commitment [151], which is based on the discrete logarithm assumption over strong-prime-order groups. The scheme is defined below.

Definition 2.4.9 (Pedersen commitment). The Pedersen commitment $C_{\text{Ped}}$ is defined as follows. $\mathcal{QR}(\mathbb{Z}_p^*)$ denotes quadratic residues in $\mathbb{Z}_p^*$.

- $\text{Setup}_{\text{Ped}}(1^\lambda)$ outputs $(p, g, y)$ where $p = 2q + 1$ is a strong $(\lambda + 1)$-bit prime, $g$ is a random generator of $G = \mathcal{QR}(\mathbb{Z}_p^*)$, and $y$ is a random element of $G$.
- $\text{Commit}_{\text{Ped}}((p, g, y), m)$ takes a message $m \in \mathbb{Z}_q$ and outputs $(c, (r, m))$ where $r \leftarrow \mathbb{Z}_q^*$ is randomly sampled and $c = g^r y^m \mod p$. 
The Pedersen commitment is unconditionally perfectly hiding since the commitment randomness \( r \) is chosen randomly in \( \mathbb{Z}_q^* \) and therefore \( c = g^r y^m \) is distributed randomly in \( G \), for any message \( m \). The computational binding property follows from the hardness of discrete logarithm.

**Succinct perfectly hiding commitments.** Parameter sizes in the standard Pedersen commitment grow with the message size, and in particular, the commitment is not succinct. We can obtain a commitment scheme that is both succinct and perfectly hiding by first creating a Pedersen commitment \( c' \) and then computing its hash using a sufficiently compressing collision-resistant hash function.

### 2.5 Encryption

We use the notation \( E = (\text{Gen}, \text{Enc}, \text{Dec}) \) for *public-key* encryption (PKE) schemes and \( \text{SKE} = (\text{SGen}, \text{SEnc}, \text{SDec}) \) for *secret-key* encryption (SKE) schemes. These primitives are described further in the following subsections.

#### 2.5.1 Public-key encryption

**Definition 2.5.1 (PKE (syntax)).** A public-key encryption scheme over a message space \( \mathcal{M} = \{ M_\lambda \}_{\lambda \in \mathbb{N}} \) is a tuple of PPT algorithms \( E = (\text{Gen}, \text{Enc}, \text{Dec}) \) satisfying the following. Let the ciphertext space be denoted by \( \mathcal{C} = \{ C_\lambda \}_{\lambda \in \mathbb{N}} \).

- The key generation algorithm \( \text{Gen} \) takes input \( 1^\lambda \), where \( \lambda \) is the security parameter, and outputs a public key and secret key pair \( (pk, sk) \).
- The encryption algorithm \( \text{Enc} \) takes as input \( 1^\lambda \), a public key \( pk \) and a message \( m \in M_\lambda \) and outputs a ciphertext \( c \in C_\lambda \).
- The decryption algorithm \( \text{Dec} \) is a deterministic algorithm that takes as input \( 1^\lambda \), a secret key \( sk \) and a ciphertext \( c \), and outputs a decryption \( m' \in M_\lambda \).

The first input \( 1^\lambda \) to \( \text{Enc} \) and \( \text{Dec} \) is usually left implicit, as accordingly are the subscripts on \( \mathcal{M} \) and \( \mathcal{C} \).
Definition 2.5.2 (Correctness of PKE). A public-key encryption scheme \( E \) satisfies correctness if there is a negligible function \( \varepsilon \) s.t. for all \( \lambda \in \mathbb{N} \) and \( m \in M \),

\[
\Pr \left[ \begin{array}{c}
(pk, sk) \leftarrow \text{Gen}(1^\lambda) \\
 c \leftarrow \text{Enc}(pk, m) \quad : \quad m \neq m' \\
m' \leftarrow \text{Dec}(sk, c)
\end{array} \right] \leq \varepsilon(\lambda) . \tag{2.5}
\]

\( E \) moreover satisfies perfect correctness if (2.5) holds for \( \varepsilon = 0 \).

This thesis refers to two standard security definitions for public-key encryption schemes, known as CPA security and CCA security. The latter is strictly stronger than the former, and their definitions are given next.

The security notion defined next is known variously as security against chosen-plaintext attacks (CPA), security against lunchtime attacks, CPA security, or semantic security.

Definition 2.5.3 (CPA security of PKE). A public-key encryption scheme \( E \) is CPA secure if for all PPT adversaries \( A = (A_1, A_2) \), there is a negligible function \( \varepsilon \) s.t. for all \( \lambda \in \mathbb{N} \),

\[
\Pr \left[ \begin{array}{c}
(pk, sk) \leftarrow \text{Gen}(1^\lambda) \\
 (m_0, m_1) \leftarrow A_1(pk) \\
 b \leftarrow \{0, 1\} \quad : \quad b = b' \\
 c \leftarrow \text{Enc}(pk, m_b) \\
 b' \leftarrow A_1(c)
\end{array} \right] \leq 1/2 + \varepsilon(\lambda) . \tag{2.6}
\]

The security notion defined next is known as security against chosen-ciphertext attacks (CCA) or CCA security. The differences from Definition 2.5.3 (CPA security) are shown in red below: in a nutshell, the adversary is allowed additional access to decryption oracles, which it does not have in the CCA security definition.

Definition 2.5.4 (CCA security of PKE). A public-key encryption scheme \( E \) is CCA secure if for all PPT adversaries \( A = (A_1, A_2) \), there is a negligible function \( \varepsilon \) s.t. for
In the above, \( \text{Dec}(\cdot, \cdot) \) denotes an oracle defined as follows:

\[
\text{Dec}(sk, c') = \begin{cases} 
\bot & \text{if } c' = c \\
\text{Dec}(sk, c') & \text{otherwise}
\end{cases}
\]

### 2.5.2 Secret-key encryption

**Definition 2.5.5 (SKE (syntax)).** A secret-key encryption scheme over a message space \( M = \{M_\lambda\}_{\lambda \in \mathbb{N}} \) is a tuple of PPT algorithms \( \text{SKE} = (\text{SGen}, \text{SEnc}, \text{SDec}) \) satisfying the following. Let the ciphertext space be denoted by \( C = \{C_\lambda\}_{\lambda \in \mathbb{N}} \).

- The key generation algorithm \( \text{SGen} \) takes no input and outputs a secret key \( sk^* \).
- The encryption algorithm \( \text{SEnc} \) takes as input a message \( m \in M_\lambda \) and a secret key \( sk^* \), and outputs a ciphertext \( c \in C_\lambda \).
- The decryption algorithm \( \text{SDec} \) is a deterministic algorithm that takes as input a ciphertext \( c \) and a secret key \( sk^* \), and outputs a decryption \( m' \in M_\lambda \).

The first input \( 1^\lambda \) to \( \text{Enc} \) and \( \text{Dec} \) is usually left implicit, as accordingly are the subscripts on \( M \) and \( C \).

**Definition 2.5.6 (Correctness of SKE).** A secret-key encryption scheme \( \text{SKE} \) satisfies
correctness if there is a negligible function $\varepsilon$ s.t. for all $\lambda \in \mathbb{N}$ and $m \in M$,

$$\Pr \left[ \begin{array}{c}
    sk^* \leftarrow \text{SGen}(1^\lambda) \\
    c \leftarrow \text{SEnc}(sk^*, m) : m \neq m' \\
    m' \leftarrow \text{SDec}(sk^*, c)
\end{array} \right] \leq \varepsilon(\lambda). \quad (2.8)$$

SKE moreover satisfies perfect correctness if (2.8) holds for $\varepsilon = 0$.

**Definition 2.5.7** (Perfect security of SKE). A secret-key encryption scheme $\text{SKE} = (\text{SGen}, \text{SEnc}, \text{SDec})$ is perfectly secure if for all $\lambda \in \mathbb{N}$, for all messages $m_0, m_1 \in M_\lambda$, the following two distributions are identical:

$$\{\text{SEnc}(sk, m_0)\}_{sk \leftarrow \text{SGen}(1^\lambda)} = \{\text{SEnc}(sk, m_1)\}_{sk \leftarrow \text{SGen}(1^\lambda)}.$$

The notions of CPA and CCA security defined in Section 2.5.1 have analogues in the secret-key setting. In the context of secret-key encryption, CPA security is the only notion relevant for this thesis, so the definition of CCA security is omitted.

The definition of CPA security for secret-key encryption schemes follows. The security experiment is much like the public-key version, except that the adversary has an encryption oracle: this was not necessary in the public-key setting since anyone in possession of the public key can produce encryptions without any oracle.

**Definition 2.5.8** (CPA security of SKE). A secret-key encryption scheme $\text{SKE}$ is CPA secure if for all PPT adversaries $A = (A_1, A_2)$, there is a negligible function $\varepsilon$ s.t. for all $\lambda \in \mathbb{N}$,

$$\Pr \left[ \begin{array}{c}
    sk \leftarrow \text{SGen}(1^\lambda) \\
    (m_0, m_1) \leftarrow A_1^{\text{SEnc}(sk, \cdot)}(pk) \\
    b \leftarrow \{0, 1\} \\
    c \leftarrow \text{SEnc}(pk, m_b) \\
    b' \leftarrow A_2^{\text{SEnc}(sk, \cdot)}(c)
\end{array} \right] : b = b' \leq 1/2 + \varepsilon(\lambda). \quad (2.9)$$

The following definition of “pseudorandom SKE” is less standard terminology than
the preceding definitions. Another term used in the literature to this type of encryption scheme is *secret-key encryption with pseudorandom ciphertexts*. Pseudorandom secret-key encryption schemes can be built from one-way functions.

**Definition 2.5.9** (Pseudorandomness of SKE). *A secret-key encryption scheme* SKE *is pseudorandom if for any polynomial* \( t = t(\lambda) \), *and any sequence of messages* \( m_1, \ldots, m_t \), *the two following distributions are computationally indistinguishable:*

1. Sample \( sk^* \leftarrow \text{SGen}(1^\lambda) \) and output \( (\text{SEnc}(sk^*, m_1), \ldots, \text{SEnc}(sk^*, m_t)) \).
2. Output \( t \) independent samples from \( U(\mathcal{C}_\lambda) \).

### 2.6 Zero knowledge

A zero-knowledge argument\(^3\) allows a *prover* \( P \) to convince a *verifier* \( V \) of a fact without revealing any unnecessary information about the fact in the process of doing so. \( P \) can provide to \( V \) a tuple \( (R, x, \pi) \) consisting of a binary relation \( R \), an *input* \( x \), and a *proof* \( \pi \), such that the verifier is convinced that \( \exists w \text{ s.t. } (x, w) \in R \) yet cannot infer anything about the witness \( w \). Informally, three properties are required of zero-knowledge arguments: *completeness*, that any true statement can be proven by the honest algorithm \( P \) such that \( V \) accepts the proof; *soundness*, that no purported proof of a false statement (produced by *any* algorithm \( P^* \)) should be accepted by the honest verifier \( V \); and *zero-knowledge*, that the proof \( \pi \) reveals no information beyond what can be inferred just from the desired statement that \( (x, w) \in R \).

In general, zero-knowledge proofs and arguments are two-party protocols that may be interactive. For this thesis, we are interested specifically in a *non-interactive* variant of zero-knowledge argument (with a setup assumption), introduced and defined formally in the next subsection.

---

\(^3\) *Zero-knowledge proof* is a more commonly used term than *zero-knowledge argument*. The two terms denote very similar concepts; the difference is in lies only in the nature of the *soundness* guarantee (i.e., that false statements cannot be convincingly attested to), which is computational for arguments and statistical for proofs. Somewhat confusingly, the attestations produced by argument systems, as well as the attestations produced by proof systems, are both often referred to as *proofs*. 

40
2.6.1 Zero-knowledge SNARKs

“SNARK” stands for “Succinct Non-interactive ARgument of Knowledge.” Our constructions use preprocessing zero-knowledge SNARKs (zk-SNARKs) for arithmetic circuit satisfiability, following the construction of [32]. “Preprocessing” means that there is a potentially expensive algorithm (PP) that must be run as a one-time setup, on whose output the subsequent generation and verification of proofs depends. We leave implicit the term “preprocessing” from here on.

**Definition 2.6.1** (Zero-knowledge SNARK (adapted from [30])). A zk-SNARK is a triple of PPT algorithms \( \text{SNARK} = (\text{PP}, \text{Prove}, \text{Verify}) \) with the following syntax.

- **PP**\((1^\lambda, R)\) takes as input the security parameter \( \lambda \) and a description of a binary relation \( R \) (represented as an arithmetic circuit of size polynomial in \( \lambda \)), and outputs a pair \((pk_R, vk_R)\) of a proving key and verification key.

- **Prove**\((pk_R, (x, w))\) takes as input a proving key \( pk_R \) and an input-witness pair \((x, w)\) and outputs a proof \( \pi \) attesting to \( x \in L_R \), where

\[
L_R = \{ x : \exists w \text{ s.t. } (x, w) \in R \} .
\]

- **Verify**\((vk_R, (x, \pi))\) takes as input a verification key \( vk_R \) and an input-proof pair \((x, \pi)\) and outputs a bit indicating whether \( \pi \) is a valid proof for \( x \in L_R \).

The proving key and verification key need only be generated once per arithmetic circuit, and can thereafter be publicly known and reused to prove membership in the corresponding binary relation arbitrarily polynomially many times.

**SNARK** is a zk-SNARK if the following four conditions hold.

- **(Completeness)** Honestly generated proofs must verify with overwhelming prob-
ability. Formally, \( \exists \) negligible \( \varepsilon \) s.t. \( \forall \lambda \in \mathbb{N}, \forall R, \forall (x, w) \in R, \)

\[
\Pr \left[ \begin{array}{l}
    (pk_R, vk_R) \leftarrow \text{PP}(1^\lambda, R) \\
    \pi \leftarrow \text{Prove}(pk_R, (x, w)) : b = 1 \\
    b \leftarrow \text{Verify}(vk_R, (x, \pi))
\end{array} \right] \geq 1 - \varepsilon(\lambda) .
\]

- **(Perfect zero-knowledge)**\(^5\) The distribution of keys and proof reveals no information about the witness (i.e., can be simulated without the witness). Formally, there is a stateful PPT algorithm Sim = (Sim\(_1\), Sim\(_2\)) such that \( \forall R, \forall \lambda, \) and \( \forall (x, w) \in R, \) the following distributions are identical:

\[
\begin{align*}
    &\left\{ (pk_R, vk_R) \leftarrow \text{PP}(1^\lambda, R) \right. \\
    &\pi \leftarrow \text{Prove}(pk_R, (x, w)) \\
    &\text{output} (pk_R, vk_R, x, \pi) \\
\end{align*}
\equiv
\begin{align*}
    &\left\{ (pk_R, vk_R, \tau) \leftarrow \text{Sim}_1(1^\lambda, R) \right. \\
    &\pi \leftarrow \text{Sim}_2(\tau, x) \\
    &\text{output} (pk_R, vk_R, x, \pi)
\end{align*}
\]

\( \tau \), sometimes referred to as the "simulation trapdoor," is the state passed from Sim\(_1\) to Sim\(_2\). In general, \( \tau \) may be considered to contain \( (pk_R, vk_R) \) as outputted by Sim\(_1\).

- **(Succinctness)** For any \( \lambda \in \mathbb{N} \), binary relation \( R \), and \( (pk, vk) \in \text{PP}(1^\lambda, R) \), an honestly generated proof has size \( \text{poly}(\lambda) \) (but constant in other parameters), and \( \text{Verify}(vk, (x, \cdot)) \) runs in time \( \text{poly}(\lambda) \cdot O(|x|) \).

- **(Proof of knowledge)** For any PPT algorithm Prove*, there is a PPT algorithm Extract and negligible function \( \varepsilon \) s.t. \( \forall \lambda \in \mathbb{N}, \forall R, \) for any auxiliary input \( z \in \)

\(^5\)The zero-knowledge condition can be relaxed to statistical or computational variants; these definitions are standard so we omit the details of the variants. The SNARK construction of [32] was originally stated to satisfy a weaker notion of zero-knowledge than that given here, but in fact the [32] construction also satisfies the stronger definition given here, and the proofs of [32] suffice unchanged to prove the stronger definition [178]. That perfect zero-knowledge can be achieved is remarked in the appendix of [31].
{0,1}^*,

\[
\Pr \begin{bmatrix}
(pk_R, vk_R) \leftarrow \text{PP}(1^\lambda, R) \\
(x^*, \pi^*) \leftarrow \text{Prove}^*(z, pk_R, vk_R) \\
w^* \leftarrow \text{Extract}(z, pk_R, vk_R) \\
b^* \leftarrow \text{Verify}(vk_R, (x^*, \pi^*))
\end{bmatrix} : b = 1 \land (x^*, w^*) \notin R \leq \varepsilon(\lambda).
\]

Remark 2.6.2. Even if the setup PP is compromised, it is possible to ensure that only soundness is damaged (i.e., it may be possible to generate proofs of false statements), but data secrecy is still preserved [177].

2.7 Multi-party computation (MPC)

MPC allows a set of \( n \) parties, each in possession of respective private data \( x_1, \ldots, x_n \), to jointly compute the output of a function \( y = f(x_1, \ldots, x_n) \) on their private inputs. \( y \) is computed via an interactive protocol executed by the parties.

Secure MPC provides two guarantees: correctness and secrecy. Correctness means that the output \( y \) is equal to \( f(x_1, \ldots, x_n) \). Secrecy means that any adversary that corrupts some subset \( S \subset \{p_1, \ldots, p_n\} \) of the parties learns nothing about \( \{x_i : p_i \notin S\} \) beyond what can already be inferred given the adversarial inputs \( \{x_i : p_i \in S\} \) and the output \( y \). Secrecy is formalized by stipulating that a simulator that is given only \( (\{x_i : p_i \in S\}, y) \) as input must be able to produce a "simulated" protocol transcript that is indistinguishable from the actual protocol execution run with all the real inputs \( (x_1, \ldots, x_n) \). The simulation is described formally by \( \mathcal{F}_{\text{MPC}} \) below.

**Communication models.** There are two main communication models considered in the literature: broadcast and pairwise (point-to-point) channels. In the former, parties' messages are visible to all other parties; in the latter, parties send messages exclusively to specific other parties.

**Adversarial models.** Definitions of security for MPC have been formalized with respect to different types of adversaries. Typically, there is a bound on the number
Ideal Functionality 1 $\mathcal{F}_{\text{MPC}}$

In the ideal model, a trusted third party $T$ is given the inputs, computes the function $f$ on the inputs, and outputs to each player $i$ his output $y_i$. In addition, we model an ideal process adversary $S$ who attacks the protocol by corrupting players in the ideal setting.

Public parameters. $\kappa \in \mathbb{N}$, the security parameter; $n \in \mathbb{N}$, the number of parties; and $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$, the function to be computed.

Private parameters. Each player $i \in [n]$ holds a private input $x_i \in \{0, 1\}^*$.

1. **INPUT.** Each player $i$ sends his input $x_i$ to $T$.
2. **COMPUTATION.** $T$ computes $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$.
3. **OUTPUT.** For each $i \in [n]$, $T$ sends the output value $y_i$ to party $i$.
4. **OUTPUT OF VIEWS.** After the protocol terminates, each party produces an output, as follows. Each uncorrupted party $i$ outputs $y_i$ if he has received his output, or $\perp$ if not. Each corrupted party outputs $\perp$. Additionally, the adversary $S$ outputs an arbitrary function of the information that he has learned during the execution of the ideal protocol.

Let the output of party $i$ be denoted by $V_i$, and let the view outputted by $S$ be denoted by $V_S$. Let $V_{\text{ideal}}$ denote the collection of all the views:

$$V_{\text{ideal}} = (V_S, V_1, \ldots, V_n).$$

of parties an adversary may corrupt. A static adversary must choose which parties to corrupt before the protocol begins; an adaptive adversary may choose parties to corrupt during the course of protocol execution. A passive adversary sees the internal state of all corrupted parties but does not cause the corrupt parties’ actions to deviate from the protocol specification; an active adversary is stronger than a passive one, in that it can prescribe arbitrary deviations from the protocol specification, for corrupt parties. Passive adversaries are sometimes called honest-but-curious or semi-honest adversaries.

Yet other adversarial models have been proposed in the multiparty computation literature, but the ones listed above are some of the most common, and suffice for the purposes of this thesis. A formal definition of security follows.

**Definition 2.7.1** (Security of MPC). A multi-party protocol $F$ is said to securely realize $\mathcal{F}_{\text{MPC}}$ against static (resp., adaptive) passive (resp., active) adversaries making up to $t = t(n)$ corruptions if for any static (resp., adaptive) passive (resp., active)
PPT adversary $A$ attacking the protocol $F$ by corrupting a subset of players $S \subseteq [n]$ such that $|S| \leq t$, there is a PPT ideal adversary $S$ which, attacking $\mathcal{F}_{\text{MPC}}$ by corrupting the same subset $S$ of players, can output a view $\mathcal{V}_S$ such that $V_A \approx \mathcal{V}_S$, where $V_A$ is the view outputted by the real-world adversary $A$. A view is the collection of all local randomness and protocol messages seen (i.e., sent and received by corrupt parties).

If for any computationally unbounded adversary $A'$ there is a simulator $S'$ such that $V_{A'} = \mathcal{V}_{S'}$, we say the protocol perfectly securely realizes $\mathcal{F}_{\text{MPC}}$ (against the relevant type of adversary).

Different outputs for different parties. In the informal description at the top of Section 2.7, we described $n$ parties computing a single output $y = f(x_1, \ldots, x_n)$ to be learned by all parties. In contrast, the ideal functionality $\mathcal{F}_{\text{MPC}}$ described a protocol where each party receives their own output $y_i$, which could be different from other parties’ outputs. The latter definition clearly implies the former (by simply setting all the $y_i$ to be equal to $y$). In fact, the definitions are equivalent and the former implies the latter too. A protocol $F$ for computing a single global output value $y = f(x_1, \ldots, x_n)$ can be easily converted into a protocol $F'$ for releasing different values to different parties (say, $y_i = f_i(x_1, \ldots, x_n)$ for party $i$), in the following way.

1. At the start of the protocol $F'$, each party $i \in [n]$ samples a uniformly random value $\rho_i$ such that $|\rho_i| = |y|$, and treats $(x_i, \rho_i)$ as its input to the protocol $F$.
2. The protocol $F$ is run to compute the single-output functionality $f^*$ which is defined as follows:

$$f^*((x_1, \rho_1), \ldots, (x_n, \rho_n)) = (y_1 \oplus \rho_1, \ldots, y_n \oplus \rho_n),$$

where $y_i = f_i(x_1, \ldots, x_n)$ for each $i \in [n]$.
3. Upon learning the output $(z_1, \ldots, z_n)$ in the protocol $F$, each party $i \in [n]$ “un-masks” its own output value as $y_i = z_i \oplus \rho_i$.

The output $(z_1, \ldots, z_n)$ reveals nothing to any party $i$ about other parties’ outputs,
since \((z_j)_{j \neq i}\) are distributed uniformly at random in the absence of the other parties’ local randomness \((\rho_j)_{j \neq i}\).

**Possibility theorems.** The following are some seminal general possibility results for multi-party computation.

**Theorem 2.7.2 ([25, 51, 6]).** \(\mathcal{F}_{\text{MPC}}\) can be perfectly securely realized for any \(n\)-input functionality \(f\) against \(t < n/2\) static passive corruptions or \(t < n/3\) static active corruptions by a \(n\)-party protocol, assuming communication by pairwise channels. Moreover, the bound of \(t < n/3\) is tight.

**Theorem 2.7.3 ([88]).** Assuming that trapdoor permutations exist, \(\mathcal{F}_{\text{MPC}}\) can be perfectly securely realized for any \(n\)-input functionality \(f\) against up to \(t \leq n - 1\) static active corruptions.

**Guaranteed output delivery.** An additional desirable property of multi-party computation protocols, other than correctness and secrecy, is *guaranteed output delivery*: the property that every honest (non-corrupt) player is guaranteed to receive her correct output, even in the presence of an adversary. This property is known to be achievable if and only if \(t < N/2\) (that is, a majority of the players are honest) [88, 54].

### 2.8 Append-only ledgers

An *append-only ledger* is a log containing an ordered sequence of data consistently visible to anyone (within a designated system), and to which data may be appended over time, but whose contents once instated may be not be edited or deleted. The append-only nature of the ledger is important for the maintenance of a *globally consistent* and *tamper-proof* data record over time. Practical instantiations of append-only ledgers have the properties of global consistency and tamper-resistance based on certain assumptions (e.g., that not all parties with the ability to influence the ledger contents collude maliciously).
Twenty years ago, the clipper chip project which advocated that cryptographic keys for encrypted voice communications should be held in escrow by government agencies was discontinued. Today, brought to public attention by polarizing recent events such as the San Bernardino terrorist attack, the question of how (and if) access to plaintext should be made available to law enforcement agencies for encrypted digital information is back on. Technological development has wrought many important changes since twenty years ago: digital communication — encrypted or otherwise — has become prevalent and widely considered necessary to lead a normal life in the developed world; and alongside, government practices of monitoring, storing, and accessing ordinary citizens’ digital data have expanded dramatically, overreaching what many thought acceptable constitutionally or from a civil liberties perspective [167].

Regardless of an eventual solution, it seems clear that a key factor in any workable solution to address these concerns in the long run will be the ability of institutions, such as government agencies, to give credible public assurances that their practices adhere to acceptable standards, in surveillance as well as other areas. Toward this end, it is beneficial for institutions to be transparent about their practices to the maximum extent possible consistent with their ability to enforce laws, and consistent with national security. Moreover, it is important that formal procedures exist that ensure that government agencies can be held accountable under the law for their adherence to such standards, so as to provide the guarantee that violations constituting breaches of public trust (e.g., arising from abuse of power or from compromised technical infrastructure) can be detected even in principle.

Cryptography is an essential tool to design transparency and accountability solutions for complex applications. Broadly speaking, cryptography enables fine-grained control over authenticity, validity, and secrecy of data: important components of any system of transparency or accountability. Informally, by authenticity I refer to certification that some data is indeed what it purports to be; by validity I refer to the

---

6We refer to U.S. government practices in this chapter, while noting that evidence indicates that many other countries engage in similar behavior [167].
adherence of some data to application-specific constraints that must be satisfied to be considered “valid” within a particular context; and by secrecy I refer to keeping some data, or some parts of some data, provably inaccessible to some or all parties in a system.

The next two chapters present cryptographic solutions to two specific problems of institutional accountability, that are of timely relevance in the United States at the time of writing, as outlined next.

**Practical accountability of secret processes**

Chapter 3 addresses an accountability challenge faced by the U.S. federal court system in the context of electronic surveillance orders. The court system is currently exploring ways to improve the accountability of electronic surveillance, an opaque process often involving cases sealed from public view and tech companies subject to gag orders against informing surveilled users. One judge has proposed publicly releasing some metadata about each case on a paper cover sheet as a way to balance the competing goals of (1) secrecy, so the target of an investigation does not discover and sabotage it, and (2) accountability, to ensure timely unsealing of gag orders and more generally that surveillance powers are not abused.

Inspired by the courts' accountability challenge, we illustrate in this chapter how accountability and secrecy are simultaneously achievable when modern cryptography is brought to bear. Specifically, we design a system that increases transparency and accountability for one of the leading United States electronic surveillance laws, the Electronic Communications Privacy Act (ECPA) [69]. Our system improves configurability while preserving secrecy, offering new tradeoffs potentially more palatable to the risk-averse court system. Judges, law enforcement, and companies publish commitments to surveillance actions, prove in zero-knowledge that their behavior is consistent, and compute aggregate surveillance statistics by multi-party computation (MPC).

We moreover demonstrate that these primitives perform efficiently at the scale of the federal judiciary. To do so, we implement a hierarchical form of MPC that
mirrors the hierarchy of the court system. We also develop statements in succinct zero-knowledge (SNARKs) whose specificity can be tuned to calibrate the amount of information released. All told, our proposal not only offers the court system a flexible range of options for enhancing accountability in the face of necessary secrecy, but also yields a general framework for accountability in a broader class of secret information processes.

**Accountability in the face of secret laws**

Chapter 4 examines the question of whether cryptography can support accountability in the adverse — indeed, Kafkaesque\(^7\) — situation where even the very rules that institutions claim to follow are secret. We consider a scenario where certain rules are secret due to national security considerations, motivated by the documented phenomenon of classified rulings under the Foreign Intelligence Surveillance Act (FISA) [74] by the U.S. Foreign Intelligence Surveillance Court,\(^8\) which then are used as precedent justifying future also (classified) rulings. Since 9/11, journalists, scholars and activists have pointed out that secret laws — a body of law whose details and sometime mere existence is classified as top secret — were on the rise in all three branches of the U.S. government due to growing national security concerns. Amid heated current debates on governmental wishes for exceptional access to encrypted digital data, one of the key issues is: which mechanisms can be put in place to ensure that government agencies follow agreed-upon rules in a manner which does not compromise national security objectives? This promises to be especially challenging when the rules, according to which access to encrypted data is granted, may themselves be secret.

\(^7\)"Our Laws are not generally known; they are kept secret by the small group of nobles who rule us. We are convinced that these ancient laws are scrupulously administered; nevertheless it is an extremely painful thing to be ruled by laws that one does not know." — Franz Kafka, *Parables and Paradoxes* [109].

\(^8\)The FISA Court is a U.S. federal court originally created in 1978 to oversee surveillance warrants against foreign spies, but whose oversight has since expanded to much more general intelligence operations, and by today "has created a secret body of law giving the National Security Agency the power to amass vast collections of data on Americans" [125]. Their operations are classified, and the court typically hears arguments only from the government [113].
We show how the use of cryptographic protocols, and in particular, the idea of zero-knowledge proofs can ensure accountability and transparency of the government in this extraordinary, seemingly deadlocked, setting. We propose an efficient record-keeping infrastructure with versatile publicly verifiable audits that preserve (information-theoretic) secrecy of record contents as well as of the rules by which the records are attested to abide. Our protocol is based on existing blockchain and cryptographic tools including commitments and zero-knowledge SNARKs, and satisfies the properties of indelibility (i.e., no back-dating), perfect data secrecy, public auditability of secret data with secret laws, accountable deletion, and succinctness. We also propose a variant scheme where entities can be required to pay fees based on record contents (e.g., for violating regulations) while still preserving secrecy. Our scheme can be directly instantiated on the Ethereum blockchain (and a simplified version with weaker guarantees can be instantiated with Bitcoin).
Chapter 3

Practical Accountability of Secret Processes

This chapter presents results from “Practical Accountability of Secret Processes” [78], a joint work with Jonathan Frankle, Daniel Shaar, Shafi Goldwasser, and Daniel J. Weitzner that is conditionally accepted to USENIX Security 2018.

3.1 The state of electronic surveillance today

In 2016, Google received 27,850 requests from a law enforcement agencies for data implicating 57,392 user accounts [90], and Microsoft received 9,907 requests implicating 24,288 users [140]. These are some of the only publicly available figures on the scope of law enforcement requests for data from technology companies under the Electronic Communications Privacy Act (ECPA) [69], one of the leading United States electronic surveillance laws.

Underlying many of these requests is a court order. A court order is an action by a federal judge requiring a company to turn over data related to a target (i.e., a user) who is suspected of committing a crime, in response to a request from a law enforcement agency. ECPA is one of many electronic surveillance laws, and each follows somewhat different legal procedures; however, they broadly tend to follow the idealized workflow in Figure 3-1. First, a law enforcement agency presents a
surveillance request to a federal judge (arrow 1). The judge can either approve or deny it. Should the judge approve the request, she signs an order authorizing the surveillance (arrow 2). A law enforcement agency then presents this order, describing the data to be turned over, to a company (arrow 3). The company either complies or contests the legal basis for the order with the judge (arrow 4). Should the company’s challenge be accepted, the order could be narrowed (arrow 5) or eliminated; if not, the company turns over the requested data (arrow 6).

These court orders are the primary procedural marker that surveillance ever took place. They are often sealed, temporarily hidden from the public for a period of time after they are issued. In addition, companies are frequently gagged, banned from discussing the order with the target. These measures are vital for the investigative process: were a target to discover that she were being surveilled, she could potentially change her behavior, endangering the underlying investigation.

According to Judge Stephen Smith, a federal magistrate judge whose role includes adjudicating requests for surveillance, gags and seals come at a cost. Openness of judicial proceedings has long been part of the common-law legal tradition, and court documents are presumed to be public by default. To Judge Smith, a court’s public records are “the source of its own legitimacy” [165]. Judge Smith has noted several specific ways that gags and seals undermine the legal mechanisms meant to balance the powers of investigators and investigated [165]:

1. **Indefinite sealing.** Many sealed orders are ultimately forgotten by the courts
which issued them, meaning ostensibly temporary seals become permanent in practice [165]. To determine whether she was surveilled, a member of the public would have to somehow discover the existence of a sealed record, confirm the seal had expired, and request the record. Making matters worse, these records are scattered across innumerable courthouses nationwide.

2. Inadequate incentive and opportunity to appeal. Seals and gags make it impossible for a target to learn she is being surveilled, let alone contest or appeal the decision. Meanwhile no other party has the incentive to appeal. Companies prefer to reduce compliance and legal costs by cooperating. A law enforcement agency would only consider appealing when a judge denies its request; however, Judge Smith explains that even then, agencies often prefer not to “risk an appeal that could make ‘bad law’” by creating precedent that makes surveillance harder in the future [165]. As a result, judges who issue these orders have “literally no appellate guidance” [165].

3. Inability to discern the extent of surveillance. Judge Smith laments that lack of data means “neither Congress nor the public can accurately assess the breadth and depth of current electronic surveillance activity” [166]. Several small efforts shed some light on this process: wiretap reports by the Administrative Office of the U.S. Courts [171] and the aforementioned “transparency reports” by tech companies [140, 90]. These reports, while valuable, clarify only the faintest outlines of surveillance.

The net effect is that electronic surveillance laws are not subject to the usual process of challenge, critique and modification that keeps the legal system operating with the bounds of constitutional principles. This lack of scrutiny ultimate reduces public trust: we lack answers to many basic questions. Does surveillance abide by legal and administrative rules? Do agencies present authorized requests to companies, and do companies return the minimum amount of data to comply? To a public concerned about the extent of surveillance, credible assurances would increase trust. To foreign governments that regulate cross-border dataflows, such assurances could determine
whether companies have to drastically alter data management when operating abroad. Yet, today, no infrastructure for making such assurances exists.

To remedy these concerns, Judge Smith proposes that each order be accompanied by a publicly available cover sheet containing general metadata about an order (e.g., kind of data searched, crimes suspected, length of the seal, reasons for sealing) [166]. The cover sheet would serve as a visible marker of sealed cases; when a seal expires, the public can hold the court accountable by requesting the sealed document. Moreover, the cover sheet metadata enables the public to compute aggregate statistics about surveillance, complementing the transparency reports released by the government and companies.

Designing the cover sheet involves balancing two competing instincts: (1) for law enforcement to conduct effective investigations, some information about surveillance must be hidden and (2) public scrutiny can hold law enforcement accountable and prevent abuses of power. The primary design choice available is the amount of information to release.

Our contribution. As a simple sheet of paper, Judge Smith’s proposal is inherently limited. Inspired by Judge Smith’s proposal, we demonstrate the accountability achievable when the power of modern cryptography is brought to bear. Cryptographic commitments can indicate the existence of a surveillance document without revealing its content. Secure multi-party computation can allow judges to compute aggregate statistics about all cases — information currently scattered across voluntary transparency reports — without revealing data about any particular case. Zero-knowledge arguments can demonstrate that a particular surveillance action (e.g., requesting data from a company) follows properly from a previous surveillance action (e.g., a judge’s order). All of this information is stored on an append-only ledger, giving the courts a way to release information and the public a definitive place to find it. Courts can post additional information to the ledger, from the date that a seal expires to the entirety of a cover sheet. Together, these primitives facilitate a flexible accountability strategy that can provide greater assurance to the public while protecting the secrecy
of the investigative process.

To show the practicality of these techniques, we evaluate MPC and zero-knowledge protocols that amply scale to the size of the federal judiciary.\(^1\) To meet our efficiency requirements, we develop a hierarchical MPC protocol that mirrors the structure of the federal court system. Our implementation supports sophisticated aggregate statistics (e.g., "how many judges ordered data from Google more than ten times?"), and scales to hundreds of judges who may not stay online long enough to participate in a synchronized multiround protocol. We also implement succinct zero-knowledge statements about the consistency of data held in different commitments; the legal system can tune the specificity of these statements in order to calibrate the amount of information released. Our implementations extend the existing libraries Webmpc [38, 120] and Jiff [108] (for MPC) and LibSNARK [160] (for zero-knowledge).

Finally, we observe that the federal court system’s accountability challenge is an instance of a broader class of secret information processes, where some information must be kept secret among participants (e.g., judges, law enforcement, and companies) engaging in a protocol (e.g. surveillance as in Figure 3-1), yet the propriety of the participants’ interactions are of interest to an auditor (e.g., the public). After presenting our system as tailored to the case study electronic surveillance, we describe a framework that generalizes our strategy to any accountability problem that can be framed as a secret information process. Concrete examples include clinical trials, public spending, and other surveillance regimes.

**Organization of the rest of the chapter.** Section 3.2 discusses related work. Section 3.3 introduces the system design of our accountability scheme for the court system, and Section 3.4 presents detailed protocol algorithms. Sections 3.5 and 3.6 discuss the implementation and performance of hierarchical MPC and succinct zero knowledge. Section 3.7 generalizes our framework to a range of scenarios beyond electronic surveillance, and Section 3.8 concludes.

\(^1\)There are approximately 900 federal judges [1].
3.2 Related work

Accountability. The term *accountability* has many definitions. [73] categorizes technical definitions of accountability according to the timing of interventions, information used to assess actions, and response to violations; [72] further formalizes these ideas. [147] surveys definitions from both computer science and law. [187] surveys definitions specific to distributed systems and the cloud.

In the terminology of these surveys, our focus is on *detection* (“The system facilitates detection of a violation” [73]) and *responsibility* (“Did the organization follow the rules?” [147]). Our additional challenge is that we consider protocols that occur in secret. Other accountability definitions consider how “violations are tied to punishment” [73, 119]; we defer this question to the legal system and consider it out of the scope of this work. Unlike [150], which advocates for “prospective” accountability measures like access control, our view of accountability is entirely retrospective.

Implementations of accountability in settings where remote computers handle data (e.g., the cloud [150, 169, 170] and healthcare [131]) typically follow the transparency-centric blueprint of *information accountability* [184]: remote actors record their actions and make logs available for scrutiny by an auditor (e.g., a user). In our setting (electronic surveillance), we strive to release as little information as possible subject to accountability goals, meaning complete transparency is not a solution.

Cryptography and government surveillance. Kroll, Felten, and Boneh [118] also consider electronic surveillance but focus on cryptographically ensuring that participants only have access to data when legally authorized. Such access control is orthogonal to our work. Their system includes an audit log that records all surveillance actions, but much of the logged data is encrypted with a “secret escrow key” to access it. In contrast, motivated by a problem articulated directly by the legal community, we focus exclusively on accountability and develop a more nuanced and fully public framework for a general class of accountability problems of which electronic surveillance is one instance.

Other research that suggests applying cryptography to enforce rules governing
access-control aspects of surveillance includes: [110], which considers NSA telephony metadata surveillance; [162], which uses private set intersection for surveillance involving joins over large databases; and [161], which uses the same technique for searching communication graphs. The work of [89] suggests succinct zero-knowledge could be used to achieve accountability for “secret laws,” (rather than “secret processes”), i.e., where the regulations being followed may themselves be secret.

**Efficient MPC and zk-SNARKs.** LibSNARK [160] is the primary existing implementation of zk-SNARKs. More numerous implementation efforts have been made for MPC under a range of assumptions and adversary models, e.g., [38, 120, 108, 5, 182, 58].

### 3.3 System design

We present the design of our proposed system for accountability in electronic surveillance. Section 3.3.1 informally introduces four cryptographic primitives and their security guarantees.2 Section 3.3.2 outlines the configuration of the system — where data is stored and processed. Section 3.3.3 describes the workflow of the system in relation to the surveillance process summarized in Figure 3-1. Section 3.3.4 discusses the packages of design choices available to the court system, exploiting the flexibility of the cryptographic tools to offer a range of options that trade off between transparency and accountability.

#### 3.3.1 Cryptographic tools

We briefly overview the roles of the cryptographic tools that our system will use.

**Append-only ledgers.** In our system, the ledger records credibly time-stamped information about surveillance events. Typically, data stored on the ledger will cryp-

---

2For rigorous formal definitions of these cryptographic primitives, we refer to any standard cryptography textbook (e.g., [112]).
Cryptographic commitments. In our system, commitments indicate that a piece of information (e.g., a surveillance order) exists and that its content can credibly be opened at a later time. Posting commitments of the ledger also establishes the existence of a piece of information at a given point in time.

Zero-knowledge. In our system, zero-knowledge makes it possible to reveal how secret information relates to a system of rules or to other pieces of secret information without revealing any further information. Concretely our implementation (detailed in Section 3.6) allows law enforcement to attest (1) knowledge of the content of a commitment \( c \) (demonstrating the ability to later open \( c \)); and (2) that the content of a commitment \( c \) to an identifier in a surveillance request is equal to the content of a prior commitment \( c' \) by a judge issuing a surveillance order. In case even (2) reveals too much information, our implementation supports not specifying \( c' \) exactly, and instead attesting that \( c' \) lies in a given set \( S \) (e.g., \( S \) could include all judges' surveillance authorizations from the last six months).

Secure multi-party computation. In our system, MPC enables computation of aggregate statistics about the extent of surveillance across the entire court system, without the need to pool the sensitive data of individual judges in the clear.

Together, these cryptographic tools support three capabilities necessary to assure
accountability:

1. **Trusted records of events:** The append-only ledger and cryptographic commitments create a trustworthy record of surveillance events without revealing sensitive information to the public.

2. **Demonstration of compliance:** Zero-knowledge arguments allow parties to provably assure the public that relevant rules have been followed without revealing any secret information.

3. **Aggregate statistics without handling secrets:** MPC enables analysts to accurately compute and release statistics about surveillance events without ever accessing sensitive information of individual parties.

### 3.3.2 System configuration

Our system is centered around a publicly visible, append-only ledger where the various entities involved in the electronic surveillance process can post information. As depicted in Figure 3-2, every judge, law enforcement agency, and company contributes data to this ledger. Judges post cryptographic commitments to all orders issued. Law enforcement agencies post commitments to their activities (warrant requests to judges and data requests to companies), and zero-knowledge arguments about the requests they issue. Companies do the same for the data they deliver to agencies. Members of the public can view and verify all data posted to the ledger.

Each judge, law enforcement agency, and company will need to maintain a small amount of infrastructure: a computer terminal through which to compose posts and local storage (the ovals in Figure 3-2) to store sensitive information (e.g., the content of sealed court orders). To attest to the authenticity of posts to the ledger, each participant will need to maintain a private signing key and publicize a corresponding verification key. We assume that public-key infrastructure could be established by a reputable party like the Administrative Office of the Courts.
Figure 3-3: Data posted to the public ledger as the protocol runs. Time moves from left to right. Each rectangle is a post to the ledger. Dashed arrows between rectangles indicate that the source of the arrow could contain a visible reference to the destination. The ovals contain the entities that make each post.

3.3.3 Workflow

Posting to the ledger. The workflow of our system augments the electronic surveillance workflow in Figure 3-1 with additional information posted to the ledger — mostly in the form of cryptographic commitments — as depicted in Figure 3-3. When a judge issues an order (step 2 of Figure 3-1), she also posts a commitment to the order and additional metadata about the case. At a minimum, this metadata must include the date that the order’s seal (if any) expires; depending on the system configuration, she could post other metadata (e.g., Judge Smith’s cover sheet). The commitment allows the public to later verify that the order was properly unsealed, but reveals no information about the commitment’s content in the meantime.

The agency then uses this order to request data from a company (step 3 in Figure 3-1) and posts a commitment to this request alongside a zero-knowledge argument that the request is compatible with a court order (and possibly also with other legal requirements). If the company responds with matching data (step 6 in Figure 3-1), it posts a commitment to its response and an argument that it furnished only the data implicated by the order.

This system does not require commitments to all actions in Figure 3-1. For example, it only requires a law enforcement agency to commit to a successful request for data (step 3) rather than any proposed request (step 1). The system could easily be augmented with additional commitments and proofs as desired by the court system.
The zero-knowledge arguments about relationships between commitments reveal one additional piece of information. For a law enforcement agency to prove that its committed data request is compatible with a particular court order, it must reveal which specific committed court order authorized the request. In other words, the zero-knowledge arguments reveal the links between specific actions of each party (dashed arrows in Figure 3-3). These links could be eliminated, reducing visibility into the workflow of surveillance. Instead, entities would argue that their actions are compatible with some court order among a group of recent orders.

**Aggregate statistics.** At configurable intervals, the judges use MPC to compute aggregate statistics about their surveillance activities. An analyst, such as the Administrative Office of the Courts, receives the result of this MPC and posts it to the ledger. The particular aggregate statistics computed are at the discretion of the court system. They could include figures already tabulated in the Administrative Office of the Court's Wiretap Reports [171] (orders by state and by criminal offense) and in company-issued transparency reports [90, 140] (requests and number of users implicated by company over six-month intervals). Due to the generality of MPC, it is theoretically possible to compute any function of the information known to each of the judges. For performance reasons, we restrict our focus to totals and aggregated thresholds, a set of operations expressive enough to replicate existing transparency reports.

The statistics themselves are calculated using MPC. In principle, the hundreds of magistrate and district court judges could attempt to directly perform MPC with each other. However, as we find in Section 3.5, computing even simple functions among hundreds of parties is prohibitively slow. Moreover, the logistics of getting every judge online simultaneously with enough reliability to complete a multiround protocol would be difficult; if a single judge went offline, the protocol would stall.

Instead, we compute aggregate statistics in a hierarchical manner as depicted in Figure 3-4. We exploit the existing hierarchy of the federal court system. Each of the lower-court judges is under the jurisdiction of one of twelve circuit courts of appeals.
Each lower-court judge computes her individual component of the larger aggregate statistic (e.g., number of orders issued against Google in the past six months) and divides it into twelve secret shares, sending one share to (a server controlled by) each circuit court of appeals. Distinct shares are represented by separate colors in Figure 3-4. So long as at least one circuit server does not reveal its shares, the lower-court judges can be assured — by the security of the secret-sharing scheme — that their contributions to the larger statistic are confidential. The circuit servers engage in a twelve-party MPC that reconstructs the judges' input data from the shares, computes the desired function, and reveals the result to the analyst. By concentrating the computationally intensive and logistically demanding part of the MPC process in twelve stable servers, this design eliminates many of the performance and reliability challenges of the flat (non-hierarchical) protocol (Section 3.5 discusses performance).

3.3.4 Additional design choices

The preceding section described our proposed system with its full range of accountability features. This configuration is only one of many possibilities. Although cryptography makes it possible to release information in a controlled way, the fact remains that revealing more information poses greater risks to investigative integrity. Depending on the court system’s level of risk-tolerance, features can be modified or removed
entirely to adjust the amount of information disclosed.

**Cover sheet metadata.** A judge might reasonably fear that a careful criminal could monitor cover sheet metadata to detect surveillance. At the cost of some transparency, judges could post less metadata when committing to an order. (At a minimum, the judge must post the date at which the seal expires.) Although the cover sheets are integral to Judge Smith’s proposal, they were designed to supply the information necessary to assess the scale of surveillance. MPC replicates this outcome without releasing information about individual orders.

**Commitments by individual judges.** In some cases, posting a commitment might reveal too much. In a low-crime area, mere knowledge that a particular judge approved surveillance could spur a criminal organization to change its behavior. To mitigate this concern, judges could delegate the responsibility of posting to the ledger to the same judicial circuits that mediate the hierarchical MPC. Doing so would mask information about the individual judges that approved the surveillance.

**Aggregate statistics.** The aggregate statistic mechanism offers a wide range of choices about the data to be revealed. For example, if the court system is concerned about revealing information about individual districts, statistics could be aggregated by any number of other parameters, including the type of crime being investigated or the company from which the data was requested.

### 3.4 Protocol definition

We now define a complete protocol capturing the workflow from Section 3.3. We assume a public-key infrastructure and synchronous communication on authenticated point-to-point channels.

**Preliminaries.** The protocol is parametrized by:

- a secret-sharing scheme \((\text{Share}, \text{Recon})\),
• a commitment scheme \( C \),
• a zk-SNARK SNARK,
• a multi-party computation protocol \( MPC \), and
• a function \( \text{CoverSheet} \) that maps court orders to cover sheet information.

Several parties participate in the protocol:

• \( n \) judges \( J_1, \ldots, J_n \);
• \( m \) law enforcement agencies \( A_1, \ldots, A_m \);
• \( q \) companies \( C_1, \ldots, C_q \);
• \( r \) trustees \( T_1, \ldots, T_r \); and
• \( P \), a party representing the public.

• \( \text{Ledger} \), a party representing the public ledger;
• \( \text{Env} \), a party called “the environment,” which models the occurrence over time of exogenous events.

\( \text{Ledger} \) is a simple ideal functionality allowing any party to (1) append entries to a time-stamped append-only ledger and (2) retrieve ledger entries. Entries are authenticated except where explicitly anonymous.

\( \text{Env} \) is a modeling device that specifies the protocol behavior in the context of arbitrary exogenous event sequences occurring over time. Upon receipt of message \( \text{clock} \), \( \text{Env} \) responds with the current time. To model the occurrence of an exogenous event \( e \) (e.g., a case in need of surveillance), \( \text{Env} \) sends information about \( e \) to the affected parties (e.g., a law enforcement agency).

The specification of parties’ behavior follows.

---

3In our specific case study, \( r = 12 \) and the trustees are the twelve U.S. Circuit Courts of Appeals. The trustees are the parties which participate in the multi-party computation of aggregate statistics based on input data from all judges, as show in Figure 3-4 and defined formally later in this subsection.

4For the purposes of our exposition, this could be an arbitrary judge. In practice, it would likely depend on the jurisdiction in which the surveillance event occurs, and in which the law enforcement agency operates, and perhaps also on the type of case.

5For simplicity of exposition, Algorithm 1 only addresses the case \( d \neq \text{reject} \), and omits the possibility of appeal by the agency. The algorithm could straightforwardly be extended to encompass appeals, by incorporating the decision to appeal into \( A_i^{dp} \).
Algorithm 1 Law enforcement agency $A_i$

- On receipt of a **surveillance request event** $e = (\text{Surveil}, u, s)$ from Env, where $u$ is the public key of a company and $s$ is the description of a surveillance request directed at $u$: send message $(u, s)$ to a judge.\(^4\)
- On receipt of a **decision message** $(u, s, d)$ from a judge where $d \neq \text{reject}$:\(^5\)(1) send request $(s, d)$ to company $u$; (2) generate a commitment $c = \text{Commit}((s, d); \rho)$ to the request and store $(c, s, d, \rho)$ locally; (3) generate a SNARK proof $\pi$ attesting compliance of $(s, d)$ with relevant regulations; (4) post $(c, \pi)$ to the ledger.
- On receipt of a **data revelation request** $(c, z)$ from the public: generate decision $b \leftarrow A_i^{dp}(c, z)$. If $b = \text{accept}$, send to the public $P$ the message and randomness $(m, \rho)$ corresponding to $c$; else, send $(c, z, b)$ to a judge.\(^4\)
- On receipt of an **audit request** $(c, P, z)$ from the public: generate decision $b \leftarrow A_i^{dp}(c, P, z)$. If $b = \text{accept}$, generate a SNARK proof $\pi$ attesting compliance of $(s, d)$ with the regulations indicated by the audit request $(c, P, z)$; else, send $(c, P, z, b)$ to a judge.\(^4\)
- On receipt of a **data revelation order** $(d, c, z)$ from a judge: if $d = \text{accept}$, send to the public $P$ the message and commitment randomness $(m, \rho)$ corresponding to $c$.
- On receipt of an **audit order** $(d, c, P, z)$ from a judge: if $d = \text{accept}$, generate a SNARK proof $\pi$ attesting compliance of $(s, d)$ with the regulations indicated by the audit request $(c, P, z)$.

Algorithm 2 Judge $J_i$

- On receipt of a **surveillance request** $(u, s)$ from an agency $A_j$: (1) generate decision $d \leftarrow J_i^{dp1}(s)$; (2) send response $(u, s, d)$ to $A_j$; (3) generate a commitment $c = \text{Commit}((u, s, d); \rho)$ to the decision and store $(c, u, s, d, \rho)$ locally; (4) post (CoverSheet$(d), c)$ to the ledger.
- On receipt of **denied audit or data revelation request information** $\zeta$ from an agency $A_j$: generate decision $d \leftarrow J_i^{dp2}(\zeta)$, and send $(d, \zeta)$ to $A_j$ and to the public $P$.

Algorithm 3 Company $C_i$

- Upon receiving a **surveillance request** $(s, d)$ from an agency $A_j$, if the court order $d$ bears the valid signature of a judge, then: (1) reply to $A_j$ by furnishing the requested data $\delta$; (2) generate commitment $c = \text{Commit}(\delta; \rho)$ and store $(c, \delta, \rho)$ locally; (3) generate a SNARK proof $\pi$ attesting that $\delta$ is compliant with a judge-signed order $d$; (4) post $(c, \pi)$ anonymously to the ledger.

67
Algorithm 4 Trustee $T_i$

- Upon receiving an **aggregate statistic event message** $e = (\text{Compute}, f, D_1, \ldots, D_n)$ from Env:
  1. For each $i' \in [r]$ (such that $i' \neq i$), send $e$ to $T_{i'}$.
  2. For each $j \in [n]$, send the message $(f, D_j)$ to $J_j$. Let $\delta_{j,i}$ be the response from $J_j$.
  3. With parties $T_1, \ldots, T_r$, participate in the MPC protocol $\text{MPC}$ with input $(\delta_{1,i}, \ldots, \delta_{n,i})$, to compute the functionality $\text{ReconInps} \circ f$, where $\text{ReconInps}$ is defined as follows.

$$\text{ReconInps}((\delta_{1,i'}, \ldots, \delta_{n,i'}))_{i' \in [r]} = (\text{Recon}(\delta_{j,1}, \ldots, \delta_{j,r}))_{j \in [n]}$$

Let $y$ denote the output from the MPC.\(^6\)

4. Send $y$ to $J_j$ for each $j \in [n]$.\(^7\)

- Upon receiving an **MPC initiation message** $e = (\text{Compute}, f, D_1, \ldots, D_n)$ from another trustee $T_{i'}$:
  1. Receive a secret-share $\delta_{j,i}$ from each judge $J_j$ respectively.
  2. With parties $T_1, \ldots, T_r$, participate in the MPC protocol $\text{MPC}$ with input $(\delta_{1,i}, \ldots, \delta_{n,i})$, to compute the functionality $\text{ReconInps} \circ f$.

**Agencies.** Each agency $A_i$ has an associated decision-making process $A_i^{dp}$, modeled by a stateful algorithm that maps public data revelation requests to $\text{accept} \cup \{0, 1\}^*$, where the output either is an acceptance of the request, or a description of why the agency chooses to deny it. Each agency operates according to Algorithm 1, which is parametrized by the individual agency’s $A_i^{dp}$. (In practice, we assume that $A_i^{dp}$ would be instantiated using the agency’s human decision-making process.)

**Judges.** Each judge $J_i$ has two associated decision-making processes, $J_i^{dp1}$ and $J_i^{dp2}$. $J_i^{dp1}$ maps surveillance requests to either a rejection or an authorizing court order; and $J_i^{dp2}$ maps denied requests (for audits or data revelation) to either a confirmation of the denial, or a court order overturning the denial. Each judge operates according to Algorithm 2, which is parametrized by the individual judge’s $(J_i^{dp1}, J_i^{dp2})$. 

68
The public. The public $P$ exhibits one main type of behavior in our model: upon receiving an event message $e = (a, \xi)$ from Env (describing either an audit request or a data revelation request), $P$ sends $\xi$ to agency $a$. Additionally, the public periodically checks the ledger for validity of posted SNARK proofs, and take steps to flag any non-compliance detected (e.g., through the court system or the news media). Due to the simplicity of the behavior of the public, we do not present it formally as an algorithm.

Companies and trustees. Algorithms 3 and 4 describe the behavior of companies and trustees. Companies execute judge-authorized instructions and log their actions by posting commitments on the ledger. Trustees run MPC to compute aggregate statistics based on data provided in secret-shared form by judges; MPC events (i.e., computations of aggregate statistics by trustees) are triggered by Env.

3.5 Evaluation of MPC implementation

In our proposal, judges use secure multi-party computation (MPC) to compute aggregate statistics about the extent and distribution of surveillance. Although in principle, MPC can support secure computation of any function of the judges’ data, full generality can come with unacceptable performance limitations. In order that our protocols scale to hundreds of federal judges, we narrow our attention to two kinds of functions that are particularly useful in the context of surveillance.

The extent of surveillance (totals). Computing totals involves summing values held by the parties without revealing information about any value to anyone other than its owner. Totals become averages by dividing by the number of data points. In the context of electronic surveillance, totals are the most prevalent form of computation on government and corporate transparency reports. How many court orders were approved for cases involving homocide, and how many for drug offenses? How long was the average order in effect? How many orders were issued in California? [171] Totals make it possible to determine the extent of surveillance.
The *distribution of surveillance* (thresholds). Thresholding involves determining the number of data points that exceed a given cut-off. How many courts issued more than ten orders for data from Google? How many orders were in effect for more than 90 days? Unlike totals, thresholds can reveal selected facts about the distribution of surveillance, i.e., the circumstances in which it is most prevalent. Thresholds go beyond the kinds of questions typically answered in transparency reports, offering new opportunities to improve accountability.

To enable totals and thresholds to scale to the size of the federal court system, we implemented a hierarchical MPC protocol as described in Figure 3-4, whose design mirrors the hierarchy of the court system. Our evaluation shows the hierarchical structure reduces MPC complexity from quadratic in the number of judges to linear.

We implemented protocols that make use of totals and thresholds using existing JavaScript-based MPC libraries, WebMPC [38, 120] and Jiff [108]. We opted for JavaScript to facilitate integration into a web application, which is suitable for federal judges to submit information through a familiar browser interface, regardless of the differences in their local system setups. We simulated the effect of multiple parties at the expected scale of our application, by running each party in a separate process on a computer with 16 CPU cores and 64GB of RAM. We tested these protocols on randomly generated data containing values in the hundreds, the same order of magnitude as data present in existing transparency reports. Our implementations were crafted with two design goals in mind:

1. Protocols should scale to roughly 1,000 parties, the approximate size of the federal judiciary [1], performing efficiently enough to facilitate periodic transparency reports.
2. Protocols should not require all parties to be online regularly or concurrently.

In the subsections that follow, we describe and evaluate our implementations in light of these goals.
3.5.1 Computing totals in WebMPC

WebMPC is a JavaScript-based library that can securely compute sums in a single round. Our WebMPC implementation is a relatively straightforward application of the existing library, and we see that unsurprisingly, performance scales linearly with the number of parties. We have included it as baseline benchmark, and because in specific applications where computations consisting of totals and averages are sufficient, a simple one-round MPC like WebMPC can be a cleaner solution than one with unnecessary additional machinery to compute more complex aggregate functions.

The security of the WebMPC protocol is based on the assumption that there are two parties participating in the MPC that do not collude with one another: namely, an analyst who distributes masking information to all protocol participants at the beginning of the process and receives the final aggregate statistic, and a facilitator who aggregates this information together in masked form. In more detail, the workflow of WebMPC is as follows. The participants use the masking information from the analyst to mask their inputs and send them to the facilitator, who aggregates them and sends the result (i.e., a masked sum) to the analyst. The analyst removes the mask and uncovers the aggregated result. Once the participants have their masks, they receive no further messages from any other party; they can submit this masked data to the facilitator in an uncoordinated fashion and go offline immediately afterwards. Even if some anticipated participants do not send data, the protocol can still run to completion with those who remain.

To make this protocol feasible in our setting, we need to identify a facilitator and analyst such that it is widely considered reasonable to believe they will not collude. In many circumstances, it would be acceptable to rely on reputable institutions already present in the court system, such as the circuit courts of appeals, the Supreme Court, or the Administrative Office of the Courts.

Although this protocol’s simplicity limits its generality, it also makes it possible for the protocol to scale efficiently to a large number of participants. As Figure 3-5 illustrates, the protocol scales linearly with the number of parties. Even with 400
parties — the largest size we tested — the protocol still completed in just under 75 seconds. Extrapolating from the unsurprising linear trend, it would take about three minutes to compute a summation across the entire federal judiciary. Since these statistics are likely to be computed at a monthly cadence, it is reasonable to invest three minutes of computation (or less than a fifth of a second per judge) for each statistic.

3.5.2 Thresholds and hierarchy with Jiff

To make use of MPC operations beyond totals, we turned to Jiff, another MPC library implemented in JavaScript. Jiff is designed to support MPC for arbitrary functionalities, although inbuilt support for some more complex functionalities are still under development at the time of writing. Most importantly for our needs, Jiff supports thresholding and multiplication in addition to sums. We evaluated Jiff on three different MPC protocols: totals (as with WebMPC), additive thresholding (i.e., how many values exceeded a specific threshold?), and multiplicative thresholding (i.e., did all values exceed a specific threshold?). In contrast to computing totals via summation, certain operations like thresholding require more complicated computation and multiple rounds of communication. By building on Jiff with our hierarchical MPC implementation, we demonstrate that these operations are viable at the scale required by the federal court system.

As a baseline, we ran sums, additive thresholding, and multiplicative thresholding benchmarks with all judges as full participants in the MPC protocol, a configuration we term the flat protocol (in contrast to the hierarchical protocol we present momentarily). Figures 3-6d illustrates that the running time of these protocols grows quadratically with the number of judges participating. These running times quickly
became untenable. While summation took several minutes among hundreds of judges, both thresholding benchmarks could barely handle tens of judges in the same time envelopes. These graphs illustrate the substantial performance disparity between summation and the much slower thresholding.

In Section 3.3, we described an alternative MPC configuration to reduce this quadratic growth to linear growth. As depicted in Figure 3-4, each lower-court judge splits a piece of data into twelve secret shares — one for each circuit court of appeals. These shares are sent to the corresponding courts, who conduct a twelve-party MPC that performs a total or thresholding operation based on the input shares. If \( n \) lower-court judges participate, the protocol is tantamount to computing \( n \) twelve-party summations followed by a single \( n \)-input summation or threshold. As \( n \) increases, the amount of work scales linearly, down from the quadratic growth present in the flat versions of these protocols. So long as at least one circuit court remains honest, the secrecy of the lower court data endures, by the security of the secret-sharing scheme.

Figure 3-6c illustrates the linear scaling of the twelve-party portion of the hierarchical protocols; we measured only the computation time after the circuit courts received all of the additive shares from the lower courts. While the flat summation protocol took nearly eight minutes to run on 300 judges, the hierarchical summation scaled to 1000 judges in less than 20 seconds, besting even the WebMPC results. Although thresholding characteristically remained much slower than summation, the hierarchical protocol scaled to nearly 250 judges in about the time that it took the
flat protocol to run on 35 judges. Since the running times for the threshold protocols were in the tens of minutes for large benchmarks, then linear trend is noiser than for the total protocol. Most importantly, both of these protocols scaled linearly, meaning that — given sufficient time — thresholding could scale up to the size of the federal court system. This performance is acceptable if a few choice thresholds are computed on a monthly basis. Each threshold computation required a few seconds per lower court judge.

One additional benefit of the hierarchical protocols is that lower courts do not need to stay online while the protocol is executing, a goal we articulated at the beginning of this section. A lower court simply needs to send in its shares to the requisite circuit courts, after which it is free to disconnect. In contrast, the flat protocol grinds to a half if even a single judge goes offline. The availability of the hierarchical protocol relies on a small set of circuit courts who could invest in more robust infrastructure.

3.6 Evaluation of zk-SNARKs

Recall Definition 2.4.1 (zero-knowledge SNARK), which defines a specific type of zk-SNARK that our construction relies on: namely, preprocessing SNARKs. With this type of SNARK, before participants can create and verify SNARK proofs, they must establish: a proving key, which any participant can use to create a proof, and a corresponding verification key, which any participant can use to verify a proof so created. Both of these keys are publicly known. The keys are distinct for each circuit (representing an NP relation) about which proofs are generated, and can be reused to produce as many different proofs, with respect to that circuit, as desired. Key generation uses randomness that, if known or biased, could allow participants to create proofs of false statements [29]. The key generation process must therefore protect and then destroy this information.

Using MPC to do key generation based on randomness provided by many different parties provides the guarantee that as long as at least one of the MPC participants behaved correctly (i.e., did not bias his randomness, and destroyed it afterward),
the resulting keys are good (i.e., do not permit proofs of false statements). This approach has been used in the past, most notably by the cryptocurrency Zcash [191]. Despite the strong guarantees provided by this approach to key generation when at least one party is not corrupted, concerns have been expressed about the wisdom of trusting in the assumption of one honest party in the Zcash setting, which involves large monetary values and a system design inherently centered around the principles of full decentralization.

For our system, we propose key generation be done in a one-time MPC among several of the traditionally reputable institutions in the court system, such as the Supreme Court or Administrative Office of the Courts, ideally together with other reputable parties from different branches of government. In our setting, the use of MPC for SNARK key generation does not constitute as pivotal and potentially risky a trust assumption as in Zcash, in that the court system is close-knit and inherently built with the assumption of trustworthiness of certain entities within the system. In contrast, a decentralized cryptocurrency (1) must, due to its distributed nature, rely for key generation on MPC participants that are essentially strangers to most others in the system; and (2) could be said to derive its very purpose from not relying on the trustworthiness of any small set of parties.

We note that since key generation is a one-time task for each circuit, we can tolerate a relatively performance-intensive process. Proving and verification keys can be distributed on the ledger.

3.6.1 Argument types

Our implementation supports three types of arguments.

**Argument of knowledge for a commitment** ($P_k$). Our simplest type of argument attests the prover’s knowledge of the content of a given commitment $c$, i.e., that he could open the commitment at a later time if required. Whenever a party publishes a commitment, he can accompany it with a SNARK attesting that he knows the message and randomness that were used to generate the commitment. Formally,
this is an argument that the prover knows \( m \) and \( \rho \) that correspond to a publicly known \( c \) such that \( \text{Open}(m, c, \rho) = 1 \).

**Argument of commitment equality (\( P_{eq} \)).** Our second type of argument attests that the content of two publicly known commitments \( c_1, c_2 \) is the same. That is, for two publicly known commitments \( c_1 \) and \( c_2 \), the prover knows \( m_1, m_2, \rho_1, \) and \( \rho_2 \) such that \( \text{Open}(m_1, c_1, \rho_1) = 1 \land \text{Open}(m_2, c_2, \rho_2) = 1 \land m_1 = m_2 \).

More concretely, suppose that an agency wishes to release relational information — that the identifier (e.g., email address) about which it requested data is the same identifier that a judge approved. The judge and law enforcement agency post commitments \( c_1 \) and \( c_2 \) respectively to the identifiers they used. The law enforcement agency then post an argument attesting that the two commitments are to the same value.\(^8\) Since circuits use fixed-size inputs, an argument implicitly reveals the length of the committed message. To hide this information, the law enforcement agency can pad each input up to a uniform length.

\( P_{eq} \) may be too revealing under certain circumstances: for the public to verify the argument, the agency (who posted \( c_2 \)) must explicitly identify \( c_1 \), potentially revealing which judge authorized the data request and when.

**Existential argument of commitment equality (\( P_3 \)).** Our third type of commitment allows decreasing the resolution of the information revealed, by proving that a commitment’s content is the same as that of some other commitment among many. Formally, it shows that, for publicly known commitments \( c, c_1, ..., c_N \) respectively to secret values \( (m, \rho), (m_1, \rho_1), ..., (m_N, \rho_N) \), \( \exists i \) such that \( \text{Open}(m, c, \rho) = 1 \land \text{Open}(m_i, c_1, \rho_i) = 1 \land m = m_i \). We treat \( i \) as an additional secret input, so that, for any value of \( N \), only two commitments need to be opened. This scheme trades off between resolution (number of commitments) and efficiency, a question we explore.

\(^8\)To produce a proof for \( P_{eq} \), the prover (e.g., the agency) needs to know both \( \rho_2 \) and \( \rho_1 \), but in some cases \( c_1 \) (and thus \( \rho_1 \)) may have been produced by a different entity (e.g., the judge). Publicizing \( \rho_1 \) is unacceptable as it compromises the hiding of the commitment content. To solve this problem, the judge can include \( \rho_1 \) alongside \( m_1 \) in secret documents that both parties possess (e.g., the court order).
Although we have chosen these three types of arguments to implement, LibSNARK supports the attestation of arbitrary predicates in principle, and there are likely others that would both be useful and run efficiently in practice. In particular, a useful generalization of $P_{eq}$ and $P_3$ would be to replace equality with more sophisticated, domain-specific predicates as appropriate: instead of showing that the messages $m_1, m_2$ corresponding to a pair of commitments are equal, it would be possible to attest that $p(m_1, m_2) = 1$ for other useful predicates $p$ (e.g., “greater-than” or “signed by the same court”).

### 3.6.2 Implementation

We implemented these zero-knowledge arguments with LibSNARK [160], a C++ library for creating general-purpose SNARKs from arithmetic circuits. We implemented commitments using the SHA256 hash function; $\rho$ is a 256-bit random string appended to the message before it is hashed. In this section, we show that useful statements can be proven within a reasonable performance envelope. We consider six criteria: the size of the proving key, verification key, and proof statement and the time to generate keys and create and verify proofs. We evaluated these metrics with messages from 16 to 1232 bytes on $P_k$, $P_{eq}$, and $P_3$ ($N = 100, 400, 700$, and $1000$, large enough to obscure links between commitments) on a computer with 16 CPU cores and 64GB of RAM.

**Argument size.** The argument is always just 287 bytes. Accompanying each argument are its public inputs (in this case, commitments). Each commitment is 256 bits; LibSNARK stores each bit in a 32-bit integer, meaning that an argument involving $k$ commitments takes about $1024k$ bytes of space. A bit-vector representation would save a factor of 32. An auditor needs to store these commitments anyway as part

---

9Certain other hash functions may be more amenable to representation as arithmetic circuits, and thus more “SNARK-friendly.” We opted for a proof of concept with SHA256 as it is so widely used.
of the ledger, and each commitment must be stored just once no matter how many times it is used for proofs.

**Verification key size.** The size of the verification key is proportional to the size of the circuit and its public inputs. The key was 10.6KB for $P_k$ (one commitment as input and one SHA256 circuit) and 20.83KB for $P_{eq}$ (two commitments and two SHA256 circuits). Although $P_3$ computes SHA256 just twice, its smallest input, 100 commitments, is 50 times as large as that of $P_k$ and $P_{eq}$; the keys are correspondingly larger and grow linearly with the input size. For 100, 400, 700, and 1000 commitments, the verification keys were respectively 1.0MB, 4.1MB, 7.1MB, and 10.2MB. Since only one verification key is necessary for each circuit, these keys are easily small enough to make large-scale verification feasible.

**Proving key size.** The proving keys are much larger: in the hundreds of megabytes. They increase linearly in size with the size of the circuit, so longer messages (which require more SHA256 computations), more complicated circuits, and (in the case of $P_3$) more inputs lead to larger keys. Figure 3-7a reflects this trend. Proving keys are largest for $P_3$ with 1000 inputs on 1232KB messages and shrink as the message size and the number of commitments decrease. $P_k$ and $P_{eq}$, which have simpler circuits, still have bigger proving keys for bigger messages. Although these keys are large, only entities that create each kind of proof need to store the corresponding key. Storing one key for each type of argument we have presented takes only about 1GB at the largest input sizes.
Key generation time. Key generation time increased linearly with the size of the keys, from a few seconds for $P_k$ and $P_{eq}$ on small messages to a few minutes for $P_3$ on the largest parameters (Figure 3-7b). Since key generation is a one-time process to add a new kind of proof in the form of a circuit, we find these numbers acceptable.

Argument generation time. Argument generation time increased linearly with proving key size and ranged from a few seconds on the smallest keys to a couple of minutes for largest (Figure 3-7c). Since proof generation is a one-time task for each surveillance action and the already existing administrative processes corresponding to each surveillance action often take hours or days, we find this cost acceptable.

Argument verification time. Verifying $P_k$ and $P_{eq}$ on the largest message took only a few milliseconds. Verification times for $P_3$ were larger and increased linearly with the number of input commitments. For 100, 400, 700, and 1000 commitments, verification took 40ms, 85ms, 243ms, and 338ms on the largest input. These times are still fast enough to verify many arguments quickly.

3.7 Generalization

Our proposal can be generalized beyond ECPA surveillance to encompass the broader class of secret information processes. We broadly consider situations in which independent institutions need to act in a coordinated but secret fashion and, at the same time, are subject to public scrutiny. They should be able to convince outside observers that their actions are consistent with relevant rules. As with electronic surveillance, accountability requires the ability to attest to compliance without revealing sensitive information.

Example 1 (FISA court). Accountability is needed in other electronic surveillance arenas. The U.S. Foreign Intelligence Surveillance Act (FISA) regulates surveillance in national security investigations. Because of the sensitive interests at stake, the entire process is overseen by a U.S. court that meets in secret. The tension between secrecy
and public accountability is even more extreme for the FISA court: much of the data collected through this surveillance system may stay permanently hidden inside U.S. intelligence agencies, while data collected under ECPA, which may eventually be used in public criminal trials. This opacity may be justified, but it has engendered skepticism. The public has no way of knowing what the court is doing, nor any means of assuring itself that the intelligence agencies under the authority of FISA are even complying with the rules of that court. The FISA court itself has voiced concern about that it has no independent means of assessing compliance with its orders because of the extreme secrecy involved. Applying our proposal to the FISA court, both the court and the public could receive proofs of documented compliance with FISA orders, as well as aggregate statistics on the scope of FISA surveillance activity to the full extent possible without incurring national security risk.

**Example 2 (Clinical trials).** Accountability mechanisms are also important to assess the behavior of private parties in situations far removed from electronic surveillance, such as clinical trials for new drugs. There are many parties to clinical trials and much of the information in such trials is either private or proprietary. Yet, regulators and the public have a need to know that responsible testing protocols are being observed. Our system can achieve the right balance of transparency, accountability and respect for privacy of those involved in the trials.

**Example 3 (Public fund spending).** Accountability in spending of taxpayer money is naturally a subject of public interest. Certain portions of public funds may be allocated for sensitive purposes (e.g., to defense and intelligence agencies), and the amounts and allocation of such money may be unavailable to the public due to their sensitivity. Our accountability system would enable credible public assurances that taxpayer money is being spent in accordance with stated principles, while preserving secrecy of information considered sensitive.
3.7.1 Generalized framework

We present abstractions describing the generalized version of our system and briefly outline how the concrete examples fit into this framework. A secret information process includes the following components.

- A set of participants interact with each other. In our ECPA example, these are judges, law enforcement agencies, and companies.
- The participants engage in a protocol (e.g., to execute the procedures for conducting electronic surveillance). The protocol messages exchanged are hidden from the view of outsiders (e.g., the public), and yet it is of public interest that the protocol messages exchanged adhere to certain rules.
- A set of auditors (distinct from the participants) seeks to audit the protocol, by verifying that a set of accountability properties are met.

Abstractly, our system allows the controlled disclosure of four types of information.

Existential information reveals the existence of a piece of data, be it in a participant’s local storage or the content of a communication between participants. In our case study, existential information is revealed with commitments, which indicate the existence of a document.

Relational information describes the actions participants take in response to the actions of others. In our case study, relational information is represented by the zero-knowledge arguments that attest that actions were taken lawfully (e.g., in compliance with a judge’s order).

Content information is the data in storage and communication. In our case study, content information is revealed through aggregate statistics via MPC and when documents are unsealed and their contents made public.

Timing information is a by-product of these other information. In our case study, timing information could include the time of issuance of an order, the turnaround time between a data request and its fulfilment by a company, and the date of expiry of a seal.

Revealing combinations of these four types of information with the specified cryp-
tographic tools provides the flexibility to satisfy a range of application-specific accountability properties. We outline how Examples 1, 2, and 3 can be framed as secret information processes.

**Example 1 (FISA court).** Participants are the FISA Court judges, the agencies requesting surveillance authorization, and any service providers involved in facilitating said surveillance. The protocol encompasses the legal process required to authorize surveillance, together with the administrative steps that must be taken to put surveillance in place. Auditors are the public, the judges themselves, and possibly Congress. Desirable accountability properties are similar to those in our original case study of electronic surveillance: e.g., attestations that certain rules are being followed in the issuance of surveillance orders, and release of aggregate statistics on surveillance activities conducted under FISA.

**Example 2 (Clinical trials).** Participants are the institutions (companies or research centers) conducting clinical trials, comprising scientists, ethics boards, and data analysts; the organizations that manage regulations regarding clinical trials, such as the National Institutes of Health (NIH) and the Food and Drug Administration (FDA) in the U.S.; and hospitals and other sources through which trial participants are drawn. The protocol encompasses the administrative process required to approve a clinical trial, and the procedure of gathering participants and conducting the trial itself. Auditors are the public, the regulatory organizations such as the NIH and FDA, and possibly professional ethics committees. Desirable accountability properties include, e.g., attestations that appropriate procedures are respected in recruiting participants and administering trials; and release of aggregate statistics on the results of clinical trials without compromising individual participants’ medical data.

**Example 3 (Public fund spending).** Participants are Congress (who appropriates the funding), defense and intelligence agencies, and service providers contracted in the spending of allocated funding. The protocol encompasses the processes by which Congress allocates funds to agencies, and agencies allocate funds to particular
expenses. Auditors are the public and Congress. Desirable accountability properties include, e.g., attestations that services bought were within a reasonable margin of market prices, or that they were satisfying documented needs; and release of aggregate statistics on the proportion of allocated money used, and possibly breakdowns of broad categories of spending.

3.8 Conclusion

We have presented a cryptographic answer to the accountability challenge currently frustrating the federal court system. By leveraging cryptographic commitments, zero-knowledge proofs, and secure multi-party computation, we provide the electronic surveillance process a series of scalable, flexible, and practical measures for improving accountability while maintaining secrecy. While we have focused on the case study of electronic surveillance, these accountability strategies are equally applicable to a range of other secret information processes that must be accountable to an outside auditor.

Acknowledgements for work presented in this chapter. We are grateful to Judge Stephen Smith for discussion and insights from the perspective of the U.S. court system; to Andrei Lapets, Kinan Dak Albab, Rawane Issa, and Frederick Joossens for useful discussion about Jiff and WebMPC; and to Madars Virza for helpful advice about SNARKs and LibSNARK.
Chapter 4

Public Accountability in the Face of Secret Laws

This chapter presents results from “Public Accountability vs. Secret Laws: Can They Coexist?” [89], a joint work with Shafi Goldwasser that appeared in the proceedings of the 16th Workshop on Privacy in the Electronic Society (WPES), a pre-conference workshop of the ACM Conference on Computer and Communications Security (CCS), 2017.

4.1 Secret laws

The increasing governance (or lack thereof) of intelligence operations by secret laws — that is, where even the details of law itself are classified as top secret information¹ — poses major challenges to the goals of institutional accountability. The very concept of secret law seems to go against the principles of accountability and transparency:

¹Those not interested in legal subtleties may wish to skip this footnote, which provides clarification on the usage of the term “secret law” herein. In the U.S., laws themselves cannot be secret. However, in the U.S. system of common law, the details of the way that a law shall be applied are often left ambiguous by the letter of the law, and are determined instead by the opinions and decisions issued by courts as cases arise. It is these opinions and decisions that may be classified as secret — but in such cases, it is effectively the way the law is applied that is kept secret. In this paper, we refer to this phenomenon as secret law even though it is not technically the law itself that is secret, because (a) writing “laws whose application is determined by classified court proceedings” every time would be confusing and cumbersome, and (b) in alternative legal systems where the application of the law is much closer to fully specified in the law itself, such as civil law systems that are predominant in much of the world, the closest equivalent might well be a secret law.
secret laws can render entirely indistinguishable to the public a lawfully behaving
government from one that wantonly abuses power. Indeed, recent heated discussion
among legal scholars challenges whether secret law is reasonable or even constitu-
tional in the U.S.\(^2\) Our present purpose is not to delve into discussion of whether
secret law is reasonable, but to address the reality that it exists.\(^3\) Recent work in the
legal literature has begun to consider “rules of the road for governing secret law,” and
put forth pertinent accountability and transparency considerations: e.g., [157] refers
to the importance of “public notification of secret law’s creation, presumptive sunset
and publication dates.” The House of Representatives recently held a hearing on “De-
ciphering the Debate Over Encryption: Industry and Law Enforcement Perspective”
[100] in which similar and more concerns were highlighted, e.g., [183].

The time-tested old method of reliable record-keeping and periodic audits is likely
the most important and practical tool for accountability and transparency. This
is true for individuals and businesses as well as for governments, and international
organizations. The working assumption is that when record keeping is mandated by
law or regulations, it will be followed.

However, there is only so much that traditional in-house record-keeping and audits
can achieve. A main limitation is that much of the data kept in records is sensitive
information that is not appropriate for routine disclosure, whether it be trade secrets,
personal details of employees or clients or patients, or classified information pertaining
to criminal investigations or national security. This means that irresponsible record-
keeping tends to be revealed only in special cases where the disclosure of records are
compelled (e.g., audits or court-ordered requests) rather than as a matter of routine.
Public verifiability, which is desirable to achieve the goal of transparency, is usually
out of the question in those special cases where records are checked. Another related
limitation is that of timing and back-dating: even for those records which are later
revealed (e.g., to some appropriate authority such as a judge), it is often impossible to

\(^2\)For example, the American Civil Liberties Union (ACLU) recently filed a motion challenging
the constitutionality of the secrecy of FISA court’s secret rulings [159].

\(^3\)We refer to [157] for an examination of the evidence that secret law exists (it concludes that
allegations of secret law in all three federal branches are well-founded).
be certain that the presented records are indeed the correct ones that were generated at the appropriate point in the past.

A naive approach to this problem might be to require organizations to hand over all their records as they are produced, e.g., each week or month. An immediate question is: to whom? An enormous — and, possibly misplaced — amount of trust and power would be placed in the hypothetical guardian of the sensitive internal records of all these countless organizations. Where would this vast and perpetually growing quantity of information be stored (and who would pay for it)? What if there were a hack or a leak? A breach of these records could be devastating to individuals’ privacy nationwide (and likely internationally), and could cause far-reaching damage to businesses and economies. Aside from security concerns, integrity remains an issue: would there be any way for the public to be confident that the records used in future — e.g., for audits or released under freedom of information laws — really are the same as the data that was handed over by the company to the trusted party at the appropriate point in the past? This last question is particularly concerning in the case of the records of the record-keeping organization itself: quis custodiet ipsos custodes? And yet, naturally, in matters related to national security, the government may reasonably argue that transparency along these lines would be unacceptable.

4.1.1 Our contribution: publicly auditable records on secret data and secret laws

The focus of this work is to devise a solution that enables publicly auditable record-keeping while maintaining secrecy of recorded data and the secret laws that may govern it, without a trusted third party, which is compatible with the interests of all parties, whether governmental, corporate, or otherwise.

We propose a solution that makes use of a blockchain and other cryptographic methods to simultaneously achieve the following desiderata: indelibility, secrecy, public auditability of secret data with secret laws, accountable deletion and succinctness. A natural additional desirable feature of a record-keeping scheme (which we achieve
but is of secondary importance) is to impose penalties for violations. Our construction can support penalties as well as versatile, secrecy-preserving “pricing schemes” based on record contents. This scheme can be instantiated on the Ethereum blockchain with no modification to the Ethereum protocol.

We now elaborate, informally, on each of the goals addressed.

**Timestamping and indelibility.** The creation time of a record should be public knowledge, agreed upon by all honest parties in the system. Moreover, once records are created, it should be impossible to alter them without detection at a later date (that is, back-dating records should not be feasible).

**Secrecy.** Records kept by organizations contain sensitive information. Thus, any solution for the record-keeping problem must guarantee the strongest possible secrecy respecting the contents of records. In the best of worlds, a *perfect (information-theoretic) secrecy* guarantee should be sought, which would ensure secrecy of data even against arbitrarily powerful, computationally unbounded adversaries. Indeed, one instantiation of our construction efficiently achieves perfect (information-theoretic) secrecy of sensitive data.

The secrecy requirement is of paramount importance especially in the context of national security, where the cost of failures can be catastrophic. However, it is also very important in the context of businesses which hold secret information about business practices (this is desirable since the ability to keep trade secrets can benefit the economy), and sensitive information about employees, clients, and other individuals. Release of individuals’ private information is not only an ethical issue, but also of legal significance as organizations are subject to complex regulations on handling of individuals’ data, and in particular the sharing of such data with other parties. In some (though arguably, too few) cases, explicit consent from the individuals is a prerequisite to use of their data in certain ways; an effective accountability/auditing

---

4We acknowledge that there may be certain situations in which even the creation time of a record, i.e., the time of occurrence of an unknown event, might be sensitive information that is not considered suitable to be publicly known; in such (rare) cases, our system would not be an appropriate solution and alternatives should be sought.
system must moreover be robust against reasonable individual exercise of privacy rights.

**Public auditability of secret data with secret laws.** In high-stakes settings like surveillance for national security, it is important for the public to be confident that governmental agencies are abiding by the law in their investigations. This holds even for secret law: the government should be able to assure the public, at the very least, of the fact that it is adhering to well-defined laws. This assertion in itself — that the government is behaving lawfully — is of great public interest and does not seem to pose a credible threat to national security in any circumstance. The interesting challenge that our work addresses is to devise a system allowing credible assertion by record-keepers that their recorded data adheres to specific regulations, while revealing *nothing else* about the data contents or the content of the regulations, in a provable (and information-theoretic) way.

**Accountable deletion.** Certain regulations (or an organization’s internal policies) may require that recorded data be deleted, e.g., if a user closes their account, or simply because a certain amount of time has elapsed. Data deletion events must be logged in any organization’s records, and the public audits mentioned above can be used to attest that organizations’ records show compliance with regulations about data deletion (without, of course, revealing any further details about the data). Sometimes, deleted data could include the contents of past records: accordingly, we require that upon being served with a request to reveal a particular record, an organization must always be able to *either* verifiably reveal the requested data, *or* reveal the fact that it has been legitimately deleted (as evidenced by a later record).

**Succinctness.** The volume of data that is generated by organizations is growing every day, and the amount of data that is stored for record-keeping purposes is growing alongside. While record-keeping organizations themselves must find solutions to archive their own growing quantities of data, a system for accountable record-keeping which allows auditors, and ideally also the public, to verify compliant record-keeping,
must process orders of magnitude less data. The importance of this requirement is compounded due to the number of record-keeping organizations. It would be unmanageable to process data in volumes comparable to the raw records.

**Pricing schemes.** It may be desirable to enforce fees based on the content of records. A simple example is to enforce a fine for violation of certain regulations, but payments need not necessarily be based on violations. For example, a local police department might be required to pay some fees depending on how much surveillance technology they deploy (e.g., over the course of a month), a quantity that should be logged in their records. As above, secrecy must be preserved: no information beyond the fact that the correct fee has been paid should be revealed by a secrecy-preserving pricing scheme. (In certain settings, it may be acceptable to reveal the fee amount too.)

**Overview of the proposal**

We propose an infrastructure for timestamped record-keeping on a blockchain integrated with a system for routine, publicly verifiable audits that satisfies all the above desiderata. We consider the regulatory framework *together* with the technical component to comprise a full record-keeping scheme.

Participating organizations are required to keep data records by publishing cryptographic commitments to the records in the blockchain at regular intervals. When a participating organization is later required to reveal a record — e.g., at regular intervals according to their business practice, or during an audit — they can verifiably open the commitment to the record in question. The cryptographic commitments published on the blockchain will hide the record contents, but make it impossible for the organization which published the commitment to reveal anything different from the original record committed to. Refusal to open the record, if presented with a request from appropriate authorities, will result in legal or contractual consequences (as applicable in the circumstances). Thus, the only choices for a government or a business using the infrastructure would be: reveal the record (i.e., reveal its contents
or attest to its legitimate deletion) or publicly refuse to open the commitment and face consequences.

Different types of cryptographic commitments could be used depending on the application, including ones which achieve perfect (information-theoretic) hiding. The latter would likely be the method of choice for high-stakes settings such as national security. With perfectly hiding cryptographic commitments, a commitment mathematically contains no information about the contents of the record, thus posting it on a blockchain reveals no information to an adversary regardless of computational power (except for the fact of the record’s existence). At the same time, it will be impossible for a business, government, or organization to falsify a record (i.e., reveal a different record which is consistent with the commitment) under standard cryptographic hardness assumptions. Moreover, using standard techniques, the size of the commitment to a record does not grow with the size of the record.

So far, the technical component of the scheme bears substantial similarity to existing Bitcoin-based time-stamping services (see Section 4.1.2 for more discussion of related work). Next, we discuss how to achieve the novel and richer functionality of public auditability of secret data with secret laws and pricing schemes. To achieve these, we leverage additional cryptographic tools: notably, zero-knowledge arguments.

**Secrecy-preserving audits.** In a system where commitments to records are routinely published, publicly verifiable and secrecy-preserving audits can be implemented elegantly with zero-knowledge arguments. We propose a system for routine secrecy-preserving audits based on zk-SNARKs, a particularly efficient type of zero-knowledge argument which is also publicly verifiable. zk-SNARKs have already been useful in blockchain-based systems: they are used to achieve anonymity in recently launched cryptocurrency Zcash [191, 28].5 Under our scheme, organizations are required to post zk-SNARKs on the blockchain along with the commitments to their recorded data,

5Note that zk-SNARKs require a one-time trusted setup, which can be implemented using multi-party computation between a number of carefully chosen “trustees,” and will be secure as long as not all of the trustees collude to subvert the protocol. Exactly such a trusted setup based on MPC was run in order to launch Zcash, as documented by [186]. See Section 4.3.3 for more discussion of SNARK setup.
as incontrovertible evidence that they are complying with applicable regulations. If the regulations themselves are secret, organizations must additionally publish commitments to the regulations, and prove compliance with specific regulations while keeping contents of both the regulations and the records secret. Further details are in Section 4.3.3.

**Putting a price on records.** Building upon the SNARK-based scheme, fines for non-compliance with regulations can be implemented automatically via transactions on the blockchains. This variation can be extended to support versatile system of fees (not necessarily fines) based on arbitrary properties of committed records. For example, a local police department might be required to pay some fees depending on how much surveillance technology they deploy, a quantity that should be logged in their records (and the fees could even go back to the local community). Using SNARKs, this can be implemented in a way that reveals nothing more than the amount of the fee paid and the fact that it is the correct fee amount for the committed record. Moreover, as above, compliance with fee-paying regulations can be asserted in a publicly verifiable manner even when the regulations themselves are secret. Further details are in Section 4.3.3.

Our scheme would be able to be implemented on the Ethereum blockchain with no modification to the Ethereum protocol, satisfying all of the above properties simultaneously. In the body of the paper, we discuss how this instantiation would work, as well as discussing: a simplified scheme (satisfying indelibility, secrecy, and accountable deletion but without audits) that could be implemented on the Bitcoin blockchain with no modification to the Bitcoin protocol; and an enhanced scheme that also hides fee amounts paid in a pricing scheme, and could be deployed on a hypothetical version of Zcash/Ethereum that combines Zcash’s anonymity features with Ethereum’s flexible scripts and transaction formats.⁶

---

⁶While we believe fee amounts would not be sensitive information in many settings, the ability to hide amounts could potentially enable (discussion of) more controversial pricing schemes, e.g., pay-per-wiretap.
The general framework for publicly accountable record-keeping on secret data and regulations may find utility in other domains as well. To illustrate this point, we briefly describe one potential such application.

**Whistleblowers.** Public timestamping schemes could also serve as a tool for timestamped record-keeping to lend credibility to whistleblowers. Potential whistleblowers could commit to their data in a timely fashion and record the commitments on the blockchain, either with the explicit intent of later revealing it, or “just in case.” A common tactic used against whistleblowers is to cast doubt upon their authenticity or even their sanity, and the presence of a timestamped trail of evidence could be a helpful tool to boost their credibility. Moreover, we believe that the presence of infrastructure for “just-in-case whistleblowing” could cause an interesting and beneficial shift in the incentives of corrupt organizations and individuals therein. Currently, “good” individuals in corrupt organizations are often incentivized to keep quiet for fear of retribution. Though such individuals may not take the initiative to blow the whistle, it is strictly beneficial for them to anonymously post commitments to whistleblowing data in the blockchain, which they can come out and reveal in case of any future investigation in which the corruption is brought to light or the cause for fear is removed (e.g., due to change of employer). In the long run, perhaps this landscape of individual incentives could incentivize organizations to rein in corruption.

Finally, some remarks are in order.

**Remark 4.1.1.** While it remains possible that an organization could simply refuse to open their commitments, this is analogous to the long-existing reality that an entity served with a subpoena can choose either to comply or face punishment under the law. In comparison to the existing system, our proposal provides stronger guarantees on the authenticity and public verifiability of records, and better incentivizes organizations to keep regular records as required by the law.

**Remark 4.1.2.** Our system does not provide a way to check that all generated records are committed on the blockchain. That is, a government or business could choose not
to comply with regulations and not keep records. This is also the case today.

**Remark 4.1.3.** A corrupt organization could record false information in its logs. This is true when they are required to keep traditional/paper logs, and also true in the blockchain-based record-keeping scheme that we propose. Existing measures to deter and mitigate corruption, such as audits, could be applied to blockchain-based record-keeping too.

**Remark 4.1.4.** The question of what should be done if an instance of non-compliance is discovered by way of an accountable record-keeping scheme is beyond the scope of this paper, and is likely to be context-dependent. For example, when it is desirable that the identity of the creator of a record not be known, then it is problematic for the public to directly approach the creator of the non-compliant record with a complaint; in such cases, one solution could be to designate a specific (judicial) committee to receive and arbitrate complaints.

### 4.1.2 Relation to existing services

The potential of blockchains for timestamping documents has long been recognized. A number of websites offer a timestamping service on the Bitcoin blockchain (and some also on other blockchains, notably that of Ethereum), wherein users submit documents of their choice to be timestamped, upon which the service creates a transaction containing the hash of the document and adds it to the Bitcoin blockchain. Some other services provide more complex frameworks for timestamped data management with the aim of selling their services as a generic or special-purpose data management solution to organizations. Features marketed include "proof of existence"; assurance of data "privacy", "integrity", "attribution", and "auditability"; and "scalability" to large volumes of data. To our knowledge, the auditability claimed by existing systems refers only to audits where the underlying data is revealed in an audit. We are not aware of existing proposals which put forward an inter-organizational regulatory framework (like our work), rather than targeting data management solutions within organizations. The desiderata for the two cases overlap to a degree,
but have ultimately different goals: e.g., latency is a major concern of many existing services, and is not a limiting factor for us; whereas information-theoretic security and publicly verifiable audits on secret data with secret laws are paramount in our setting and not addressed by existing solutions. More detailed discussion of existing services is given in Section 4.4, after we detail our construction.

4.2 Preliminaries

4.2.1 Blockchains

In this subsection, we give a brief overview of the high-level structure of a blockchain — a specific type of public ledger — with a stylized focus on the properties of blockchains that are relevant to this work.

A blockchain is a finite sequence of blocks $\mathcal{B} = (B_1, B_2, \ldots)$ maintained by a decentralized network of parties, in the form of a public, append-only ledger whose contents all the parties are guaranteed to agree upon. Each block $B_i$ contains some data including a set of transactions. In Bitcoin, the transactions typically record monetary transfers of the form “Alice transfers $x$ bitcoins to Bob.” More generally, a blockchain-based record-keeping system may instead support non-monetary “transactions” that serve to record certain data.

The entities that create blocks are typically called “miners,” and we will refer to the entities that create transactions as “participants.” Participants need not be miners, and vice versa. In order to have a transaction $\tau$ added to the blockchain, a participant broadcasts $\tau$ to the entire network of miners (we refer to this action as “sending $\tau$ to the blockchain network”); a miner who is creating a block will insert into the block some number of broadcasted transactions that have not yet been added to the blockchain. The integrity of the data on a blockchain depends on a good fraction of miners behaving honestly according to the protocol specification.

---

7In fact, Bitcoin’s scripting language, Script, allows for more complex monetary transfers that impose conditions upon the recipients, e.g., “Alice transfers $x$ bitcoins to anyone who can find a SHA-256 preimage of value $y$.”
**Transaction fees.** Transactions that are added to the blockchain may be accompanied by an optional transaction fee, i.e., a small amount of currency that is designated to be transferred to the party who creates a block containing the transaction in question and successfully adds that block to the blockchain. Having a transaction fee is usually advantageous for the transaction creator, as it incentivizes miners to include the transaction (quickly) in the blockchain. The default presence of a nominal transaction fee is also important to prevent denial-of-service attacks by adversaries flooding the network with transactions (sometimes called “penny-flooding”).

**Record transactions.** We describe our proposed scheme in the context of an abstract blockchain that includes a special transaction type called a record transaction. A record transaction does not entail any monetary transfer, and can store some arbitrary data \( \delta \) of the transaction creator’s choice. A record transaction \( \text{RecordTx}(pk, \delta) = (\zeta, \delta) \) contains the data \( D \) as well as any auxiliary information \( \zeta \) that is required by the transaction format of the abstract blockchain. While we recognize that Bitcoin transactions can be used to instantiate our scheme, we prefer to first present our scheme in terms of an abstract blockchain since the underlying ideas are not Bitcoin-specific.

**On the use of blockchains vs. other types of ledgers.** Though it is presented as using a blockchain and we give discussion of instantiations based on Bitcoin and Ethereum, our main construction could also be instantiated based on other types of append-only ledgers, including ones which are not fully decentralized. Such options might be preferable in situations where a small set of stakeholders can plausibly be trusted not to collude together to falsify records. Building atop an existing decentralized blockchain such as Bitcoin or Ethereum can have the disadvantage that the miners on whom the integrity of the blockchain’s content depends are typically unknown entities whose motives may be harder to ascertain than those of known

---

8Indeed, although the transaction fee is technically optional in Bitcoin, in practice, transactions with fees below 0.00001 BTC are typically discarded as spam (according to https://en.bitcoin.it/wiki/Transaction_fees).
reputable stakeholders.

An extension feature of ours — specifically, the system for automatic penalties for violations — relies more on the “smart contract” capability of the Ethereum blockchain specifically. For such a feature, use of an existing blockchain-based cryptocurrency might be advantageous in order to enforce penalties in a currency with exogenous monetary value.

4.3 Record-keeping scheme

We begin by presenting a basic framework in which it is possible for organizations to cryptographically commit to data, and to post these commitments to a public ledger. We then overview the different types of parties in our record-keeping system, and the roles they play. After that, we proceed to describe the full-fledged version of our system which incorporates zero-knowledge proofs that the committed data adheres to certain regulations.

4.3.1 Committing to records

Algorithm 5 Commit to a record

Public parameters:

- Blockchain $\mathcal{B}$.
- Succinct commitment scheme $C = (\text{Setup, Commit, Open})$.

Input: $(pk, D)$ defined as follows:

- Public key $pk$ of data owner/record creator.
- Data $D$ to be committed.

1. Generate public parameters $pp \leftarrow \text{Setup}(1^\lambda)$.
2. Generate a succinct commitment $c \leftarrow \text{Commit}(pp, D; \rho) \in \{0, 1\}^\lambda$.
3. Generate a record transaction $\tau = \text{RecordTx}(pk, c)$.
4. Send $\tau$ to the blockchain network (with an appropriate transaction fee to ensure that $\tau$ will appear in the blockchain within the next 24 hours).
5. Store $D$ and $\rho$ locally (or on the cloud) for record-keeping purposes.

The public key $pk$ may be thought of as both an identifier of the data owner/record

97
creator and as a public verification key for others to confirm a transaction’s authorship.

The routine record-keeping procedure required of an organization is to keep records on a regular (e.g., daily) basis, and run Algorithm 5 daily to record corresponding commitments in the blockchain. If the organization is later required to reveal the records (e.g., due to a routine audit, or when presented with a warrant), then it runs the decommitment algorithm to prove that the revealed records indeed match the commitment that was added to the blockchain at the appropriate point in the past.

4.3.2 Basic regulatory framework

Parties. In the basic framework, there are 3 types of parties (each modeled as interactive Turing machines). All types of parties have read access to the blockchain, and may communicate along authenticated channels.

- Organizations $O \in \text{Org}$ that create records and insert them in the blockchain. Each $O \in \text{Org}$ has the following associated public parameters:
  - Public key $pko \in \{0, 1\}^*$.
  - Frequency parameter $\varphi_o \in \mathbb{R} \cup \{\perp\}$, indicating that $O$ commits to a record to the blockchain every $\varphi_o$ days. (Further discussion of this parameter is given below.)

- Auditors $A \in \text{Aud}$, who may initiate two types of interactions:
  - Reveal committed data
  - Request arbitration of disputes

- The Court, Court, which is an arbitrator of disputes between organizations and auditors.

The frequency parameter. By default, organizations should post record transactions to the blockchain on a regular schedule, so that the time of record posting does not reveal any sensitive information. That is, all data from a given time in-
val should be aggregated to create a single record, and an “empty” record should be created even if no new data was produced during the time interval.

In some scenarios, where the timing pattern of record generation does not constitute sensitive information, the rate of record-keeping is irrelevant. In this case, we consider the frequency parameter $\varphi_0$ to have the value $\perp$.

**Record types and format.** Each record transaction contains a commitment $c \leftarrow \text{Commit}(D)$, as described in Algorithm 5. The data $D$ that is committed to may be of one of two types: *log data* or *regulations*. Formally, a regulation is represented as an efficiently checkable predicate $p$ which takes as input some log data and evaluates to 1 if and only if certain requirements are satisfied. Regulations could represent laws or other desirable checks.

**Instantiation with Bitcoin**

Bitcoin’s OP_RETURN transaction type can store up to 80 bytes of arbitrary data of the transaction creator’s choice, and is thus suitable to serve as a “record transaction” in our basic record-keeping scheme.

### 4.3.3 Zero-knowledge proofs of compliance

**Secret log data.** The most basic type of check that could be usefully performed on committed log data is a syntactic format check. More generally, zero-knowledge proofs or arguments could be used to prove that specific regulations are being adhered to (for example, quotas not being exceeded, or patient status being recorded at the required time intervals) without revealing any information about the log data other than the fact that it adheres to these regulations. If non-interactive zero-knowledge proofs of compliance were to accompany each record transaction containing log data in the blockchain, then adherence to regulations would be publicly verifiable by anyone viewing the blockchain.
Algorithm 6 Post proof that secret log data satisfies public predicate

**PUBLIC PARAMETERS:**
- Blockchain \( \mathbb{B} \).
- Succinct commitment scheme \( C = (\text{Setup}, \text{Commit}, \text{Open}) \).
- zk-SNARK \( \text{SNARK} = (\text{PP}, \text{Prove}, \text{Verify}) \).

**INPUT:** \((pk, D, p, (pk_R, vk_R))\) defined as follows:
- Public key \( pk \) of data owner/record creator.
- Data \( D \) to be committed.
- Predicate \( p \) to be attested to.
- Key-pair \((pk_R, vk_R) \leftarrow \text{PP}(R)\) where \( R \) is a circuit accepting the following relation:
  \[
  \{((c', p'), (p', D')) : c' = \text{Commit}(D'; p') \land p'(D') = 1\}.
  \]

**ALGORITHM:**
1. Generate a succinct commitment: \( c = \text{Commit}(D; \rho) \in \{0, 1\}^\lambda \), where \( \rho \) denotes the randomness used to generate the commitment.
2. Generate a SNARK proof: \( \pi = \text{Prove}(pk_R, ((c, p), (\rho, D))) \).
3. Generate a record transaction \( \tau = \text{RecordTx}(pk, (R, vk_R, c, \pi)) \).
4. Send \( \tau \) to the blockchain network.
5. Store \( D \) locally (or on the cloud) for record-keeping purposes.

Algorithm 7 Verify that secret log data satisfies a public predicate

**PUBLIC PARAMETERS:**
- Blockchain \( \mathbb{B} \).
- Succinct commitment scheme \( C = (\text{Setup}, \text{Commit}, \text{Open}) \).
- zk-SNARK \( \text{SNARK} = (\text{PP}, \text{Prove}, \text{Verify}) \).

**INPUT:** \((pk, \tau)\) defined as follows:
- Public key \( pk \) of data owner/record creator.
- Record \( \tau = (R, vk_R, c, \tilde{c}, \pi) \).

**ALGORITHM:**
1. Let \( b = \text{Verify}(vk_R, ((c, p), \pi)) \).
2. If \( b = 0 \), send \( \tau \) to Court.
Secret regulations. To provide proof of compliance with a secret regulation \( p \), an additional step is required: organizations must first create a commitment \( \hat{c} \) to the secret regulation \( p \). Then, commitments to log data must be accompanied by a zk-SNARK attesting that the log data satisfies the unknown predicate committed to in \( \hat{c} \).

Algorithm 8 Post proof that secret log data satisfies secret predicate

Public parameters:
- Blockchain \( B \).
- Succinct commitment scheme \( C = (\text{Setup}, \text{Commit}, \text{Open}) \).
- zk-SNARK \( \text{SNARK} = (\text{PP}, \text{Prove}, \text{Verify}) \).

Input: \((pk, D, p, (pk_R, vk_R))\) defined as follows:
- Public key \( pk \) of data owner/record creator.
- Data \( D \) to be committed.
- Secret predicate \( p \) to be attested to.
- Key-pair \((pk_R, vk_R) \leftarrow \text{PP}(\bar{R})\) where \( \bar{R} \) is a circuit accepting the following relation:
  \[
  \{(c', \bar{a}', (\rho', D', \bar{p}', p')) : c' = \text{Commit}(D'; \rho') \land \bar{a}' = \text{Commit}(\bar{p'}; \bar{\rho'}) \land p'(D') = 1\}.
  \]

Algorithm:
1. Generate a succinct commitment: \( c = \text{Commit}(D; \rho) \in \{0, 1\}^\lambda \), where \( \rho \) denotes the randomness used to generate the commitment.
2. Generate a succinct commitment: \( \hat{c} = \text{Commit}(p; \hat{\rho}) \in \{0, 1\}^\lambda \), where \( \hat{\rho} \) denotes the randomness used to generate the commitment.
3. Generate a SNARK proof: \( \hat{\pi} = \text{Prove}(pk_R, ((c, \hat{c}), (\rho, D, \hat{\rho}, p))) \).
4. Generate a record transaction \( \hat{\tau} = \text{RecordTx}(pk, (R, vk_R, c, \hat{c}, \hat{\pi})) \).
5. Send \( \hat{\tau} \) to the blockchain network.
6. Store \((D, p)\) locally (or on the cloud) for record-keeping purposes.

Reusing secret predicates across different log data items. When an organization attests to the satisfaction of a single secret predicate with respect to multiple log data items, it may be desirable that it be publicly known that the predicate attested to is the same across all the log data items concerned. In this case, the commitment \( \hat{c} \) to the secret predicate \( p \) could be reused for the zk-SNARK proofs generated for multiple log data items, rather than generating a fresh commitment \( \hat{c} \) for each log
Algorithm 9 Verify that secret log data satisfies a secret predicate

PUBLIC PARAMETERS:
- Blockchain $\mathbb{B}$.
- Succinct commitment scheme $C = (\text{Setup}, \text{Commit}, \text{Open})$.
- zk-SNARK $\text{SNARK} = (\text{PP}, \text{Prove}, \text{Verify})$.

INPUT: $(pk, \tilde{\tau})$ defined as follows:
- Public key $pk$ of data owner/record creator.
- Record $\tilde{\tau} = (\bar{\tau}, vk, \hat{c}, \bar{c}, \hat{\pi})$.

ALGORITHM:
1. Let $b = \text{Verify}(vk, ((c, \hat{c}), \pi))$.
2. If $b = 0$, send $\tilde{\tau}$ to Court.

Another alternative would be to allow certain records to reference previous records' commitments to secret predicates, rather than requiring a commitment to a predicate to be included explicitly in each record transaction.

The Court. The Court's task is to enforce publicly known policies for record-keeping under its jurisdiction, encompassing: the procedure by which an auditor can bring to the notice of the Court a failure of an organization to provide valid proofs of compliance regulations; the punishment associated with such failures; the circumstances in which recorded data can be legally compelled to be revealed; the steps that a requester must take to prove that his request is legitimate under the policies (e.g., presenting a warrant from a judge); the steps that the data owner can take to challenge the legitimacy of a request under the law; the time period within which requested data is required to be produced; and the punishment associated with non-cooperation with a legitimate request to reveal the data.

Mandatory data deletion. Certain regulations (or internal policies of specific organizations) require that recorded data is deleted after a certain period of time. Data deletion events should be recorded when they occur. Thus, an organization adhering to regulations, upon being served with a request to reveal a particular past
data record, must always be able to either reveal the requested data by opening the
commitment in the corresponding record transaction, or reveal a record that that data
has been legitimately deleted, by (partially) opening a commitment in a subsequent
record transaction.

When a deletion event is included in a record, a Merkle commitment must be used
to ensure that the deletion event can be revealed while keeping the rest of the record
hidden.

**Regulations as predicates.** Not all laws can easily or succinctly be represented
in the form of a well-defined predicate that log data must satisfy. The challenges
of representing law in the form of code are interesting and have been recognized
in the legal and computer science communities (e.g., [124]); while addressing these
challenges is beyond the scope of our paper, we highlight that secret laws might be a
setting in which purposely designing laws to be able to be represented in code might
be of particular benefit, as it would facilitate the use of a secure accountability scheme
such as our proposal. We note that in some cases, it may be useful to design the log
format to include, at the very least, assertions that specific requirements have been
satisfied, even if the adherence is too complicated to check directly.

**Discussion about SNARKs**

Using recent constructions [32] of zero-knowledge SNARKs (Succinct Non-interactive
ARguments of Knowledge), proof sizes would be under 300 bytes at 128 bits of security
on megabytes of record data, for natural checks such as quotas being respected,
periodicity of certain types of logs, records being signed off by appropriate staff, or
consistency between different parts of logged data. For [32], proof verification time is
under 5ms regardless of the original program’s running time. Each zk-SNARK would
attest to the output (i.e., 0 or 1) of a specific predicate $p$ evaluated on the input data
(i.e., records) $D$. We discuss efficiency considerations in more detail shortly.
**Keeping SNARK sizes small.** Existing SNARK constructions work by converting a program to a circuit, and then generating a SNARK for the circuit. Since circuit size grows with the size of input data, the input data size can influence the succinctness of the proof and the efficiency of proof generation. Using the implementation and parameter settings of [32], megabytes of record data can be directly processed to yield proofs under 300 bytes at 128 bits of security. We believe this would be sufficient for many applications provided that organizations keep frugal plain-text logs of important records.\(^9\) For cases when it is essential to process larger amounts of data in a single transaction, the recursive proof composition technique of [52] could be used to keep the proofs succinct (though proof generation time would grow).

Another factor that influences proof size and generation time is the complexity of the (circuit representation of the) predicate itself. For natural predicates such as quotas being respected, periodicity of certain types of logs, records being signed off by appropriate staff, or consistency between different parts of logged data, the circuit complexity can be linear in the data size (with small constants\(^{10}\)), provided that “circuit-friendly” cryptographic primitives are used. Circuit-friendly constructions of collision-resistant hash functions are given by [2, 85] (and previously used in the context of SNARKs by [30]), and circuit-friendly constructions of a number of cryptographic primitives, including signature schemes, are given by [116]. Though we believe it likely that many natural predicates for routine audits would be relatively simple, we remark that in principle, our system would not be limited to such simple checks. Arbitrarily complex checks — described as predicates upon the committed data — could be performed, using recursive proof composition (as mentioned above) if necessary.

Finally, we note that [32] constructs **preprocessing** zk-SNARKs, in which the statement to be proven (or more precisely, the language to be attested to) must be known

---

\(^9\)As opposed, for example, to storing all of their employees' HTML emails and word-processed reports, etc. The latter would seem to be an excess in many cases anyway, and potentially also disadvantageous as later references to the recorded data might have to search for a needle in a haystack.

\(^{10}\)E.g., respecting quotas could be represented by “threshold” type predicates that would require just one gate per data bit; and using hash-and-sign with circuit-friendly signatures, signature verifications could be done with tens of gates per bit.
during the setup phase, and parameters established during the setup depend on the statement. This is compatible with our setting because we can consider the predicate to be an *input* to the statement to be proven, together with the data commitment. In the case of secret law, the input is just a commitment to the predicate, and the actual predicate serves as part of the witness.

Bitcoin’s OP_RETURN transaction only holds up to 80 bytes, so the Bitcoin protocol could not be used unchanged for built-in audits. A factor of three or four increase in the data storage of an OP_RETURN transaction would suffice to support a SNARK-based audit scheme alongside our basic record-keeping scheme. The last increase of the OP_RETURN storage capacity was in late 2015 (Bitcoin Core 0.11.x), when it increased two-fold from 40 to 80 bytes. However, the Ethereum blockchain inherently supports storage of larger amounts of data, and we moreover suggest the use of Ethereum with our extended record-keeping scheme due to its more versatile scripting language, which can be useful for additional features such as pricing schemes, as discussed later.

**SNARK setup.** All existing zk-SNARK constructions require some form of trusted setup, either in the form of a structured common reference string, or a common random string (this is the output of PP). The more efficient constructions, such as [32], rely on a common reference string.\(^1\) The common reference string can be generated using an efficient MPC protocol between a number of carefully chosen “trustees,” and will be secure as long as at least one trustee is honest (and not compromised) [29]. Exactly such a trusted setup based on MPC was run in order to launch the cryptocurrency Zcash ??, as documented by [186]. As noted above in Remark 2.6.2, even if the setup is compromised, it is possible to ensure that only soundness is damaged (i.e., it may be possible to generate proofs of false statements), but data secrecy is still preserved [177].

\(^{11}\)We remark that the very recent zk-SNARK construction of [27] could be considered preferable in some respects, as it relies on the weaker assumption of a common random string and works in the random oracle model without knowledge assumptions (whereas [32] requires a knowledge-of-exponent assumption). However, the proof sizes in [27] are orders of magnitude larger than those in [32].
Perfect secrecy. While most zk-SNARK descriptions in the literature refer to statistical zero knowledge, all zk-SNARK constructions can be made perfect zero knowledge by allowing for a negligible error probability [28]. This would ensure perfect (information-theoretic) secrecy of committed data.

Putting a price on records

Recall that transactions are usually accompanied by a transaction fee that is transferred to the party who mines the block containing the transaction in question. This means that there is a small cost associated with adding each record transaction to the blockchain. We propose two alternative pricing schemes which could be useful in the context of record-keeping: a “free” scheme — for use when it is in the public interest to remove the burden of paying fees for record transactions — and a “paid” scheme implementing the pricing schemes discussed in Section 4.1, using zk-SNARKs.

(Almost) free record transactions. If it were determined to be in the public interest to remove the burden of paying fees for record transactions, then adding record transactions to the blockchain could be made (almost) “free” in at least a couple of different ways, as described below.

- Fee waivers: If the benefit were universally deemed to be worthwhile by miners, then miners could simply agree to use mining software that favors record transactions, instead of (roughly) prioritizing by highest transaction fees, as is currently the norm. Note that this system could not be taken advantage of by parties interested in performing monetary transactions for lower fees, since a record transaction (or Bitcoin’s OP_RETURN transaction) does not allow for monetary transfer. However, since a truly free scheme would be open season for denial-of-service attacks, we do suggest imposing a nominal fee that is much smaller than the fee for a regular transaction. (Note that a single satoshi — the smallest transferrable denomination of Bitcoin — is worth only a few thousandths

\[\text{More precisely: OP\_RETURN allows for the transaction creator to “burn” money (i.e., render it unspendable) and to pay a transaction fee, but not to transfer money to recipients of his choice.}\]
of a U.S. cent, at the time of writing.)

- **Fee subsidies:** The simple fee-waiving scheme described above is arguably "unfair" to miners, as they have to shoulder the cost in lost transaction fees, whereas the benefit of the record-keeping system applies to the whole population (i.e., not just miners). An alternative scheme that addresses this issue would be to have the transaction fees of record transactions be subsidized by the public, whether that be all members of the blockchain network, or all taxpayers in a jurisdiction. In the former case, a candidate scheme would be that a small fraction\(^{13}\) of all transaction fees of regular transactions are set aside to be used for record transaction fees. In the latter case, a special account would be maintained by the tax authorities and funded by taxpayer money allocated for the purpose of subsidizing record transaction fees. Given a sufficiently rich scripting language for transactions, such subsidies could be automatically implemented through scripts.

**Paid record transactions.** Taking a different perspective, we believe there are many situations where it would be of public benefit for record-keepers to pay fees for logging their records. In combination with the secrecy-preserving audits described in Section 4.3.3, fees could be charged based on specific properties of the committed records.

- **Fines:** Fines for (minor) violations of regulations could be automatically imposed as follows: either the record-keeping organization must either post evidence (i.e., a SNARK) showing that their record complies with a specific regulation, or accompany the record transaction with a monetary transfer of a specified amount.
- **Routine fees:** Moreover, it would be possible to implement a versatile system of fees based on arbitrary properties of committed records. For example, a local police department might be required to pay some fees depending on how much surveillance technology they deploy (e.g., over the course of a month), a quantity that should be logged in their records.

\(^{13}\)E.g., 1%; in practice, the fraction would depend on the rate of transactions processed by the currency.
A way to implement a fee scheme, like the ones described above, would be for miners to routinely check the SNARK attached to a record transaction and only accept the transaction into the blockchain if the appropriate fee is attached. Instead of simply attesting to the binary output of a predicate on the committed data, the SNARK would attest that a particular fee \( v \) is the correct output of a fee program \( P \) evaluated on the input data \( D \). Natural fee schemes which depend on adherence to certain thresholds or count the number of records of a particular type (e.g., measuring resource usage) would, as discussed in the previous subsection, have relatively efficient circuit representations allowing for succinct proofs by direct application of [32] on megabytes of data.

Note that with a small modification to the blockchain protocol, no real trust assumption need be placed on the miners: if a careless or malicious miner includes an invalid record transaction in the blockchain, then this can be detected by any member of the network by checking the corresponding SNARK (recall that it is publicly verifiable), and the modified blockchain protocol dictates that invalid transactions be ignored. Another variant protocol could furthermore impose a monetary penalty on miners who include invalid transactions in their block (e.g., by taking away their mining reward), thus incentivizing miners to perform the required checks and refrain from polluting the blockchain with invalid transaction that will end up being ignored. We refer to [148] for technical details of how such penalties could be implemented, as they define a “penalty transaction” that is straightforward to repurpose for our setting.

**To whom are the fees paid?** The traditional answer would be: to the appropriate governmental or regulatory agency. In the setting of blockchains, the fees could also be distributed among (an appropriate subset of) the members of the blockchain network, or given back to miners. We suggest that such systems could be impactful in emphasizing direct accountability. For instance, returning to the example of the local police department which is charged for the amount of surveillance technology they use: the fees could be directly shared among all members of the locality who register
a blockchain payment address in a local registry. If the registry itself were maintained on the blockchain, then the distribution of fees could again happen automatically via scripts.

**Instantiation with Ethereum (or Zcash)**

Ethereum transactions can store larger amounts of data than Bitcoin transactions, and in particular, fit the zk-SNARKs required for our full record-keeping scheme. Ethereum script is Turing-complete, so can be used to implement the automatic payment schemes discussed above.

In specific cases where it is required to implement secret law together with secret pricing schemes where the fee amounts cannot be revealed, one would need an anonymous cryptocurrency (such as Zcash). Unfortunately, Zcash transactions and scripting currently resemble Bitcoin’s, and thus would not be suitable for deploying our full record-keeping scheme. A hypothetical extension of Zcash including more expressive scripting and transactions with enough data storage to fit zk-SNARKs would support our full scheme.

### 4.4 More discussion of relation to existing services

A number of websites offer a timestamping service on the Bitcoin blockchain (and some also on other blockchains, notably that of Ethereum), wherein users submit documents of their choice to be timestamped, upon which the service creates a transaction containing the hash of the document and adds it to the Bitcoin blockchain. Some of these services embed the hash in the transaction by setting it as the “recipient address” of a tiny monetary transfer. However, repurposing the recipient field in this way is frowned upon because it bloats the unspent transaction output (UTXO) database with bogus addresses; it is preferable to use a special-purpose transaction

---

14 E.g., a local resident could register their payment address by having the mayor’s office digitally sign it, and post the payment address and signature on the blockchain in a special “registration transaction”. This would be publicly verifiable using the verification key of the mayor’s office.

15 E.g., BitcoinProof, OriginStamp, SatoshiProof, among others.
type which can store arbitrary data without “pretending” to be a monetary transfer. Bitcoin has a special transaction type called OP_RETURN which can serve this purpose. Numerous existing services\textsuperscript{16} use Bitcoin’s OP_RETURN transaction to store hashes of data on the blockchain. Moreover, many of these services use Merkle trees to generate a single succinct hash of large quantities of data, often aggregated across different users:\textsuperscript{17} this is an important measure given the limited throughput of the blockchain. Merkle trees are a basic and versatile cryptographic tool that will be an important in our constructions too.

While many of the existing services offer a simple service allowing users to timestamp any data of their choice, some other services provide more complex frameworks for timestamped data management with the aim of selling their services as a generic\textsuperscript{18} or special-purpose\textsuperscript{19} data management solution to organizations. Features marketed include “proof of existence”; assurance of data “secrecy”, “integrity”, “attribution”, and “auditability”; and “scalability” to large volumes of data. We are not aware of existing proposals which put forward an inter-organizational regulatory framework (like our work), instead of targeting data management \textit{within} organizations as do the above examples. The desiderata for the two cases overlap but have important differences: we prioritize the design of a framework that can be compatible with the interests of all parties involved, and incorporate the regulatory framework as part of the protocol, which we then prove satisfies well-motivated security definitions. Moreover, latency is not a limiting concern, for our setting as we consider the aggregation of records over long periods of time, whereas many of the data management offerings consider it essential to have latency much faster than the growth rate of the blockchain\textsuperscript{[61]}, and that is one of the main technical challenges (perhaps \textit{the} main one) that their systems aim to address.

In contrast, the primary concern in our setting (unaddressed by existing solutions)

\textsuperscript{16}E.g., Bitproof, Eternity Wall, Factom, OpenTimestamps, Stampery, and others.
\textsuperscript{17}We remark that sharing a single Merkle hash across multiple users can violate data secrecy if done naively. This can be acceptable in cases where the timestamped data is intended to be public; and in other cases, some services appear to address the problem by encrypting the data before hashing it.
\textsuperscript{18}E.g., Factom, Stampery, Tierion, among others.
\textsuperscript{19}E.g., Dipl.me specializes in educational diplomas.
public auditability of secret information to enable proof of compliance with arbitrary regulations, even in cases where the regulations themselves are kept secret. To our knowledge, the auditability claimed by existing systems refers only to audits where the underlying data is revealed in an audit; we are not aware of any scheme that targets secrecy-preserving, publicly verifiable audits, which constitute an integral feature of our system.\(^{20}\)

**Acknowledgements for work presented in this chapter.** We are grateful for insightful and helpful conversations with Oded Goldreich, Mark Moir, Daniel Weitzner and Jonathan Frankle. We also thank Madars Virza for useful discussions about zk-SNARKs.

\(^{20}\)We note that there do exist schemes that involve proving properties of certain secret data for specific applications, such as anonymous currency transfer (e.g., Zcash [191, 28]) or preventing consensus fraud ([173]). These address substantially different use cases from our scheme, and moreover, they do not consider settings where the statement to be proven is itself secret.
Part III

Individual Empowerment In Hostile Environments
Part III focuses on the use of cryptography by individuals, rather than institutions as in Part II. Cryptography is an essential tool to empower individuals to conduct communications freely and securely, and to maintain their anonymity when desired, even in adversarial environments. Indeed, the ability of individuals to use cryptographic tools, while arguably societally insignificant on the scale of a single individual, can in aggregate sway the overall societal power balance towards individuals over traditionally powerful and resource-rich institutions to a remarkable extent. This is, for example, evidenced by the so-called “Going Dark” debate in which U.S. law enforcement is expressing its consternation over allegedly severe limitations on its investigative capabilities arising from the increasing popularity of devices encrypted by default, and end-to-end encrypted messaging systems.

The next two chapters focus respectively on variants of the two of the most fundamental concepts in (public-key) cryptography: secure communication and digital signatures. Both of these concepts date back to the 1970s: the founding period of cryptography as a research field. As described next, the next two chapters propose several new security definitions for secure undetectable communication and ring signatures, that give stronger guarantees than prior existing definitions and may be particularly useful when individuals are using encryption and/or signatures in highly adversarial environments. Each chapter moreover provides constructions that satisfy the new definitions proposed therein, with accompanying security proofs.

**Secure undetectable communication via “backdoored” encryption**

In Chapter 5, we examine the feasibility of secure and undetectable point-to-point communication in a world where governments can read all the encrypted communications of their citizens. We consider a world where the only permitted method of communication is via a government-mandated encryption scheme, instantiated with government-mandated keys. Parties cannot simply encrypt ciphertexts of some other encryption scheme, because citizens caught trying to communicate outside the government’s knowledge (e.g., by encrypting strings which do not appear to be natural language plaintexts) will be arrested. The one guarantee we suppose is that the
government mandates an encryption scheme which is semantically secure against outsiders: a perhaps reasonable supposition when a government might consider it advantageous to secure its people’s communication against foreign entities. But then, what good is semantic security against an adversary that holds all the keys and has the power to decrypt?

We show that even in the pessimistic scenario described, citizens can communicate securely and undetectably. In our terminology, this translates to a positive statement: all semantically secure encryption schemes support subliminal communication. Informally, this means that there is a two-party protocol between Alice and Bob where the parties exchange ciphertexts of what appears to be a normal conversation even to someone who knows the secret keys and thus can read the corresponding plaintexts. And yet, at the end of the protocol, Alice will have transmitted her secret message to Bob. Our security definition requires that the adversary not be able to tell whether Alice and Bob are just having a normal conversation using the mandated encryption scheme, or they are using the mandated encryption scheme for subliminal communication.

Our topics may be thought to fall broadly within the realm of steganography: the science of hiding secret communication within innocent-looking messages, or cover objects. However, we deal with the non-standard setting of an adversarially chosen distribution of cover objects (i.e., a stronger-than-usual adversary), and we take advantage of the fact that our cover objects are ciphertexts of a semantically secure encryption scheme to bypass impossibility results which we show for broader classes of steganographic schemes. We give several constructions of subliminal communication schemes under the assumption that key exchange protocols with pseudorandom messages exist (such as Diffie-Hellman, which in fact has truly random messages). Each construction leverages the assumed semantic security of the adversarially chosen encryption scheme, in order to achieve subliminal communication.

An alternate perspective on our results would be an affirmative answer to the question: is there any meaningful guarantee provided by semantic security against an adversary with the power to decrypt?
Ring signatures enhanced with repudiability and (un)claimability

Ring signatures, introduced by [154], are a variant of digital signatures which certify that one among a particular set of parties has signed a particular message, without revealing which specific party is the signer. Ring signatures can be useful, for example, to certify that some leaked information comes from a privileged set of government or company officials without revealing who the whistleblower is, or to issue important orders or directives without setting up the signing individual to be made a scapegoat for repercussions.21

Ring signatures are designed to allow anyone to attach anyone else’s name to a signature. But what guarantee does a ring signature scheme provide if a purported signatory wishes to denounce a signed message—or alternatively, if a signatory wishes to later come forward and claim ownership of a signature? Given the motivation of anonymity behind the notion of a ring signature, a natural first intuition might be that parties should be able neither to denounce or claim a signature in a convincing way. That way, even if an adversary were able to place all members of a ring under duress, he could not obtain convincing evidence of the real signer’s identity.

However, prior security definitions for ring signatures do not give a conclusive answer to this question. That is, a non-signer might be able to repudiate a signature that he did not produce, or this might be impossible. Similarly, a signer might be able to later convincingly claim that a signature he produced is indeed his own, or not. While the inability to repudiate or claim signatures is a useful guarantee, as discussed above, it is not the only reasonable guarantee imaginable, as we discuss next. In any case, however, a guarantee in one direction or the other seems more desirable than no guarantee either way.

In this work, we motivate, formalize definitions, and give constructions of the new notions of repudiable, claimable, and unclaimable signatures. (The fourth notion of unrepudiability is implied by unclaimability.)

21When it comes to national security issues, for instance, there is a well-known reluctance of law-makers to “roll back” existing laws or reduce checking or surveillance measures — even if such measures are deemed likely to be beneficial — due to the risk of ending up a scapegoat (and perhaps losing one’s job) upon any future terrorist attack.
Unclaimable ring signatures. Perhaps the most surprising of these is *unclaimability*: a natural first intuition is that meaningful notions of unclaimability might be impossible to achieve, since a signer can always remember the signing randomness (and later present it as “proof” of having produced a signature). The key insight for our definition and construction of unclaimability is that the signing randomness does not constitute a convincing claim if *anyone can produce credible signing randomness* for any signature in which they are implicated.

Interestingly, even under this definition, if the signer chooses a message to sign that is tied to her identity or a secret only she knows, then she may still be able to convince others that she was the signer. For instance, the signed message could include her name, a standard digital signature, and/or the description of a hard puzzle to which only she knows the answer.\(^{22}\) This property is rather inherent: if knowledge of the contents of the message itself at the time of signing are enough to identify the signer, then no security property on the signature scheme can enforce that the signer remains hidden. But indeed, ring signatures were not designed to provide anonymity for signers who *want* to identify themselves, but rather for those who desire anonymity. Similarly, our unclaimability definition does not guarantee unclaimability for those who *want* to identify themselves, but rather provides credibility for a signer who *wants* to later be able to claim (e.g., under duress) that she could not convincingly claim the signature even if she wanted to. In particular, even an adversary with unlimited computational power who obtains the secret keys belonging to every member of the ring and a purported signing randomness from an alleged signer, he still will not be convinced of the identity of the signer, since fake signing randomness from the right distribution can be produced for every member of the ring.

\(^{22}\) For example, a hash output of which she knows the preimage.
Chapter 5

How to Subvert Backdoored Encryption

This chapter presents results from a paper titled “How to Subvert Backdoored Encryption” [99] that I coauthored with Thibaut Horel, Silas Richelson, and Vinod Vaikuntanathan.¹

Suppose that we lived in a world where the government wished to read all the communications of its citizens, and thus decreed that citizens must not communicate in any way other than by using a specific, government-mandated encryption scheme with government-mandated keys. Even face-to-face communication is not allowed: in this Orwellian world, anyone who is caught speaking to another person will be arrested. Similarly, anyone whose communications appear to be hiding information will be arrested: e.g., if the plaintexts encrypted using the government-mandated scheme are themselves ciphertexts of a different encryption scheme. However, the one assumption that we entertain in this paper, is that the government-mandated encryption scheme is, in fact, semantically secure: this is a tenable supposition with respect to a government that considers secure encryption to be in its interest, in order to prevent foreign powers from spying on its citizens’ communications.

A natural question then arises: is there any way that the citizens would be able to communicate in a fashion undetectable to the government, based only on the semantic

¹The paper is in submission at the time of writing.
security of the government-mandated encryption scheme, and *despite the fact that the government knows the keys and has the ability to decrypt all ciphertexts*? What can semantic security possibly guarantee when the adversary has the private keys?

This question may appear to fall broadly within the realm of *steganography*: the science of hiding secret communications within other innocent-looking communications (called “cover objects”), in an undetectable way. Indeed, it can be shown that if two parties have a shared secret, then based on slight variants of existing techniques for *secret-key steganography*, they can conduct communications hidden from the government.3

However, the question of whether two parties who have never met before can conduct hidden communications is more interesting. This is related to the questions of *public-key steganography* and *steganographic key exchange* which were both first formalized by von Ahn and Hopper [179]. Public-key steganography is inadequate in our setting since exchanging or publishing public keys is potentially conspicuous and thus is not an option in our setting. All prior constructions of steganographic key exchange require the initial sampling of a public random string that serves as a public parameter of the steganographic scheme. Intuitively, in these constructions, the public random string can be thought to serve the purpose of selecting a specific steganographic scheme from a family of schemes *after* the adversary has chosen a strategy. That is, the schemes crucially assume that the adversary (the dystopian government, in our story above) cannot choose its covertex distribution as a function of the public parameter.

It is conservative and realistic to expect a malicious adversary to choose the covertex distribution *after* the honest parties have decided on their communication protocol (including the public parameters). After all, malice never sleeps [135]. Alas, we show that if the covertex distribution is allowed to depend on the communication protocol, steganographic communication is impossible. In other words, for every

---

2 We note that one could, alternatively, consider an adversary with decryption capabilities arising from possession of some sort of “backdoor.” For the purposes of this paper, we opted for the simpler and still sufficiently expressive model where the adversary’s decryption power comes from knowledge of all the decryption keys.

3 We refer the reader to Section 5 for more details.
purported steganographic communication protocol, there is a covertext distribution (even one with high min-entropy) relative to which the communication protocol fails to embed subliminal messages. The relatively simple counterexample we construct is inspired by the impossibility of deterministic extraction.

**Semantic Security to the Rescue?** However, this impossibility result does not directly apply to our setting, as our covertext distribution is restricted to be a sequence of ciphertexts (that may encrypt arbitrary messages). Moreover, the ciphertexts are semantically secure against entities that are not privy to the private keys. We define the notion of a *subliminal communication scheme* (Definition 5.2.2) as a steganographic communication scheme where security holds relative to covertext distributions that are guaranteed to be ciphertexts of some semantically secure encryption scheme. Is there a way to use semantic security to enable subliminal communication?

Our first answer to this question is negative. In particular, consider the following natural construction: first, design an extractor function \( f \); then, to subliminally transmit a message bit \( b \), sample encryptions \( c \) of a (even adversarially prescribed) plaintext \( m \) using independent randomness every time, until \( f(c) = b \). There are two reasons this idea does not work. First, if the plaintext bit \( b \) is not random, the adversary can detect this fact by simply applying the extractor function \( f \) to the transmitted covertext. Second, the government can pick an adversarial (semantically secure) encryption scheme where the extractor function \( f \) is constant on all ciphertexts; this is again similar to the impossibility of deterministic extraction.

Nevertheless, we show how to circumvent these difficulties and use the semantic security of the underlying (adversarial) encryption scheme and construct a subliminal communication scheme.

**Theorem 5.0.1** (Informal version of Theorem 5.4.1). *Under the decisional Diffie-Hellman (DDH) assumption — or any other assumption that gives rise to a key exchange protocol with messages indistinguishable from random — there is a subliminal communication scheme which allows the transmission of \( O(\log \lambda) \) many bits per ciphertext after a setup phase of \( \tilde{O}(\log \lambda) \) ciphertexts (\( \lambda \) is the security parameter).*
We then show how to improve our first construction to reduce the length of the setup phase under additional assumptions.

Overview of our construction

The first idea in our construction is implicit in essentially all the works in steganography starting from [164]: namely, to achieve subliminal communication of arbitrary messages, it is sufficient to be able to undetectably communicate uniformly randomly distributed strings of one’s choice. In other words, Alice samples a string $r$ which is randomly distributed, produces some ciphertext(s) to be sent to Bob, such that Bob is able to learn $r$ from them, and yet a PPT eavesdropper Eve who sees the entire communication transcript cannot distinguish between the following two cases:

1. Alice is indeed sending (hereafter, “embedding”) random strings to Bob, or
2. Alice is producing ciphertexts using the unmodified government-mandated encryption algorithm, without embedding such random strings.

To be more precise, the indistinguishability requirement holds for any given (adversarially specified) distribution $\mathcal{M}$ of message sequences that Alice may choose to encrypt using the government-mandated encryption scheme. Notice that this does not preclude that Eve may be able to learn $r$ and indeed, our constructions do allow an eavesdropper to learn the embedded strings. Given the ability to undetectably communicated randomly distributed strings, Alice and Bob can then embed to each other the messages of a key-exchange protocol with randomly distributed messages (such as Diffie-Hellman) to establish a shared secret, and then embed to each other ciphertexts of a secret-key encryption scheme with pseudorandom ciphertexts, using the established secret as the key.

All known constructions of such undetectable random string embedding rely on the sampling of a public random seed after the adversarial strategy is fixed. In this paper, however, we are interested in bootstrapping hidden communications from the very ground up, and we are not willing to assume that the parties start from a state where such a seed is already present.
We observe that the ability to embed randomly distributed strings of one's choice — rather than, e.g., to apply a deterministic function to ciphertexts of the government-mandated encryption scheme, and thereby obtain randomly distributed strings which the creator of the ciphertexts did not choose — is crucial to the above-outlined scheme. The notion of undetectably embedding exogenous random strings — i.e., strings that are randomly distributed outside of Alice's control, but both Alice and Bob can read them — is seemingly much weaker, and certainly cannot be used to embed key exchange messages or secret-key ciphertexts. However, we observe that this weaker primitive turns out to be achievable, for our specific setting, without the troublesome starting assumption of a public random seed. We identify a method for embedding exogenous random strings into ciphertexts of an adversarially chosen encryption scheme (interestingly, our method does not generalize to embedding into arbitrary min-entropy distributions). We then exploit this method to allow the communicating parties to establish a random seed — from which point they can proceed to embed random strings of their choice, as described above.

In building this weaker primitive, in order to bypass our earlier-described impossibility result, we extract from two ciphertexts at a time, instead of one. We begin with the following simple idea: for each consecutive pair of ciphertexts $c$ and $c'$, a single hidden (random) bit $b$ is defined by $b = f(c, c')$ where $f$ is some two-source extractor. It is initially unclear why this should work because (1) $c$ and $c'$ are encryptions of messages $m$ and $m'$ which are potentially dependent, and two-source extractors are not guaranteed to work without independence; and (2) even if this difficulty could be overcome, ciphertexts of semantically secure encryption scheme can have min-entropy as small as $\omega(\log \lambda)$ (where $\lambda$ is the security parameter) and no two-source extractor known to this day can extract from such a small min-entropy.

We overcome difficulty (1) by relying on the semantic security of the ciphertexts of the adversarially chosen encryption scheme. Paradoxically, even though the adversary knows the decryption key, we exploit the fact that semantic security still holds against the extractor, which does not have the decryption key. The inputs in our case are ciphertexts which are not necessarily independent, but semantic security im-
plies that they are computationally indistinguishable from being independent. Thus, the output of \( f(c, c') \) is pseudorandom. Indeed, when \( f \) outputs a single bit (as in our construction), the output is also statistically close to random. The crucial point here is that the semantic security of the encryption scheme is used not against the government, but rather against the extraction function \( f \).

Our next observation, to address difficulty (2), is that the ciphertexts are not only computationally independent, but they are also computationally indistinguishable from i.i.d. In particular, each pair of ciphertexts is indistinguishable from a pair of encryptions of 0, by semantic security. Based on this observation, we can use a very simple "extractor", namely, the greater-than function \( GT \). In fact, \( GT \) is an extractor with two input sources, whose output bit has negligible bias when the sources have \( \omega(\log \lambda) \) min-entropy and are independently and identically distributed (this appears to be a folklore observation; see, e.g., [17]). Because of the latter condition, \( GT \) is not a true two-source extractor by standard definitions; it is rather a "same-source extractor." It turns out that because of semantic security, a same-source extractor suffices for our setting.

By repeatedly extracting random bits from pairs of consecutive ciphertexts using \( GT \), Alice and Bob can construct a shared random string \( s \). Note that in this process, Alice and Bob generate ciphertexts using the unmodified government-mandated encryption scheme, so the indistinguishability requirement clearly holds. We stress again that \( s \) is also known to a passive eavesdropper of the communication. This part of our construction, up to the construction of the string \( s \), is presented in details in Section 5.4.1. From there, constructing a subliminal communication scheme is not hard: Alice and Bob use \( s \) as the seed of a strong seeded extractor to subliminally communicate random strings of their choice as explained in Section 5.4.2. The complete description of our protocol is given in Section 5.4.3.

**Improved constructions for specific cases**

While our first construction has the advantage of simplicity, the initial phase to agree on shared random string (using the \( GT \) function) transmits only one hidden bit per
ciphertext of the government-mandated encryption scheme. A natural question is whether this rate of transmission can be improved. We show that if the government-mandated encryption scheme is succinct in the sense that the ciphertext expansion factor is at most 2, then it is possible to improve the rate of transmission in this phase to $O(\log \lambda)$ hidden bits per ciphertext using an alternative construction based on the extractor from [64]. In other words, our first result showed that if the government-mandated encryption scheme is semantically secure, we can use it to communicate subliminally; the second result shows that if the government-mandated encryption scheme is efficient, that is even better for us, in the sense that it can be used for more efficient subliminal communication.

**Theorem 5.0.2** (Informal version of Theorem 5.5.1). *If there is a secure key exchange protocol whose message distribution is pseudorandom, then there is a subliminal communication scheme in which a shared seed is established in two exchanges of ciphertexts of a succinct encryption scheme.*

Theorem 5.0.1 exploited the specific nature of the cover object distribution in our setting (specifically, that a sequence of encryptions of arbitrary messages is indistinguishable from an i.i.d. sequence of encryptions of zero). Theorem 5.0.2 exploits an additional consequence of the semantic security of the government-mandated encryption scheme: if it is succinct, then ciphertexts are computationally indistinguishable from sources of high min-entropy (i.e., they have large HILL-entropy).

It may be possible to use more advanced two-source extractors to work with a larger class of government-mandated encryption schemes (with larger expansion factors); however, the best known such extractors have an inverse polynomial error rate [50] (whereas our construction's extractor has negligible error). Consequently, designing a subliminal communication protocol using these extractors seems to require additional ideas, and we leave this as an open problem.

Finally, we show yet another approach in cases where the distribution of "innocent" messages to be encrypted under the government-mandated encryption scheme has a certain amount of conditional min-entropy. For such cases, we construct an
alternative scheme that leverages the semantic security of the encryption scheme in a rather different way: namely, the key fact for this alternative construction is that (in the absence of a decryption key) a ciphertext appears independent of the message it encrypts. In this case, running a two-source extractor on the message and the ciphertext works. The resulting improvement in the efficiency of the scheme is comparable to that of Theorem 5.0.2.

**Theorem 5.0.3** (Informal version of Theorem 5.5.3). *If there is a secure key exchange protocol whose message distribution is pseudorandom, then there is a subliminal communication scheme:*

- for any cover distribution consisting of ciphertexts of a semantically secure encryption scheme, if the innocent message distribution $M$ has conditional min-entropy rate $1/2$, or
- for any cover distribution consisting of ciphertexts of a semantically secure and succinct encryption scheme, if the innocent message distribution $M$ has conditional min-entropy $\omega(\log \lambda)$.

*In both cases, the shared seed is established during the setup phase in only two exchanges of ciphertexts.*

We conclude this introductory section with some discussion of our results in a wider context.

**On Our Modeling Assumptions.** Our model considers a relatively powerful adversary that, for example, has the ability to choose the encryption scheme using which all parties must communicate, and to decrypt all such communications. We believe that this can be very realistic in certain scenarios, but it is also important to note the limitations that our model places on the adversary.

The most obvious limitation is that the encryption scheme chosen by the adversary must be semantically secure (against third parties that do not have the ability to decrypt). Another assumption is that citizens are able to run algorithms of their choice on their own computers without, for instance, having every computational step moni-
tored by the government. Moreover, citizens may use encryption randomness of their choice when producing ciphertexts of the government-mandated encryption scheme: in fact, this is a key fact that our construction exploits. Interestingly, secrecy of the encryption randomness from the adversary is irrelevant: after all, the adversary can always choose an encryption scheme where the encryption randomness is recoverable given the decryption key. Despite this, the ability of the encryptor to choose the randomness to input to the encryption algorithm can be exploited — as by our construction — to allow for subliminal communication.

The Meaning of Semantic Security when the Adversary Can Decrypt. In an alternate light, our work may be viewed as asking the question: what guarantee, if any, does semantic security provide against adversary in possession of the decryption key? Our results find, perhaps surprisingly, that some meaningful guarantee is still provided by semantic security even against an adversary is able to decrypt: more specifically, that any communication channel allowing transmission of ciphertexts can be leveraged to allow for undetectable communications between two parties that have never met. From this perspective, our work may be viewed as the latest in a scattered series of recent works that consider what guarantees can be provided by cryptographic primitives that are somehow “compromised” — examples of recent works in this general flavor are cited in Section 5 below.

Concrete Security Parameters. From a more practical perspective, it may be relevant to consider that the government in our hypothetical Orwellian scenario would be incentivized to opt for an encryption scheme with the least possible security level so as to ensure security against foreign powers. In cases where the government considers itself to have more computational power than foreign adversaries (perhaps by a constant factor), this could create an interesting situation where the security parameter with which the government-mandated scheme must be instantiated is below what is necessary to ensure security against the government’s own computational power.

Such a situation could be risky for citizens’ hidden communications: intuitively,
our constructions guarantee indistinguishability against the citizens’ own government between an “innocent” encrypted conversation and one which is carrying hidden subliminal messages. However, the distinguishing advantage in this indistinguishability game depends on the security parameter of the government-mandated encryption scheme. Thus, it could be that the two distributions are far enough apart for the citizens’ own government to distinguish (though not for foreign governments to distinguish). We observe that citizens cognizant of this situation can further reduce the distinguishing advantage beyond that provided by our basic construction, using the standard technique of amplifying the proximity of a distribution (which is far from random) to uniformly random, by taking the XOR of several samples from the far-from-random distribution.

Having outlined this potential concern and solution, in the rest of the paper we will disregard these issues in the interest of clarity of exposition, and present a purely asymptotic analysis.

**Open Problems.** Our work suggests a number of open problems. A natural one is the extent to which the modeling assumptions that this work makes — such as the ability of honest encryptors to use true randomness for encryption — can be relaxed or removed, while preserving the ability to communicate subliminally. For example, one could imagine yet another alternate universe, in which the hypothetical Orwellian government not only mandates that citizens use the prescribed encryption scheme, but also that their encryption randomness must be derived from a specific government-mandated pseudorandom generator.

The other open problems raised by our work are of a more technical nature and better understood in the context of the specific details of our constructions; for this reason we defer their discussion to Section 5.7.

**Other related work**

The scientific study of steganography was initiated by Simmons more than thirty years ago [164], and is the earliest mention of the term “subliminal channel” referring to the
conveyance of information in a cryptosystem’s output in a way that is different from the intended output,\(^4\) of which we are aware. Subsequent works such as [45, 141, 192] initially explored information-theoretic treatments of steganography, and then Hopper, Langford, and von Ahn [98] gave the first complexity-theoretic (secret-key) treatment almost two decades later. Public-key variants of steganographic notions — namely, public-key steganography and steganographic key exchange — were first defined by [179]. There is very little subsequent literature on public-key steganographic primitives; one notable example is by Backes and Cachin [15], which considers public-key steganography against active attacks (their attack model, which is stronger than that of [179], was also considered in [98] but had never been applied to the public-key setting).

The alternative perspective of our work as addressing the question of whether any sort of secret communication can be achieved via transmission of ciphertexts of an adversarially designed cryptosystem alone fits into a scattered series of recent works that consider what guarantees can or cannot be provided by compromised cryptographic primitives. For example, Goldreich [84], and later, Cohen and Klein [56], consider what unpredictability guarantee is achieved by the classic GGM construction [86] when the traditionally secret seed is known; Austrin et al. [10] study whether certain cryptographic primitives can be secure even in the presence of an adversary that has limited ability to tamper with honest parties’ randomness; Dodis et al. [65] consider what cryptographic primitives can be built based on backdoored pseudorandom generators; and Bellare, Jaeger, and Kane [23] present attacks that work against any symmetric-key encryption scheme, that completely compromise security by undetectably corrupting the algorithms of the encryption scheme (such attacks might, for example, be feasible if an adversary could generate a bad version of a widely used cryptographic library and install it on his target’s computer).

The last work mentioned above, [23], is actually part of the broader field of kleptography, originally introduced by Young and Yung [190, 189, 188], which is also relevant context for the present work. Broadly speaking, a kleptographic attack \(\text{\textit{uses}}\)

\(^4\)This phrasing is loosely borrowed from [190].
cryptography against cryptography" [190] — i.e., changes the behavior of a cryptographic system in a fashion undetectable to an honest user with black-box access to the cryptosystem, such that the use of the modified system leaks some secret information (e.g., plaintexts or key material) to the attacker who performed the modification. An example of such an attack might be to modify the key generation algorithm of an encryption scheme such that an adversary in possession of a "back door" can derive the private key from the public key, yet an honest user finds the generated key pairs to be indistinguishable from correctly produced ones. Kleptography has enjoyed renewed research activity since [24] introduced a formal model of a specific type of kleptographic attack called \textit{algorithm substitution attacks} (ASAs), motivated by recent revelations suggesting that intelligence agencies have successfully implemented attacks of this nature at scale. Recently, [37] formalized an equivalence between certain variants of ASA and steganography.

Our setting differs significantly from kleptography in that the encryption algorithms are public and not tampered with (i.e., adhere to a purported specification), and in fact may be \textit{known} to be designed by an adversarial party.

5.1 Preliminaries

5.1.1 Key exchange

A \textit{key-exchange protocol} $\Psi$ is a two-party protocol executed between two parties $P_0$ and $P_1$, where each party outputs a key at the end of the protocol.

Informally, the security guarantee for key-exchange protocols requires that $(T, K) \approx (T, K_s)$, where $T$ is a key-exchange protocol transcript, $K$ is the shared key established in $T$, and $K_s$ is a random unrelated key.

\textbf{Definition 5.1.1.} A key-exchange protocol $\Psi$ is a two-party protocol that must satisfy the following guarantees. Below, $(T, k_0, k_1) \leftarrow \Psi_{\lambda}$ denotes that $T$ is the full transcript of an honest execution of $\Psi$ for security parameter $\lambda$, and $k_0, k_1 \in \mathcal{X}_\lambda$ are the keys outputted by $P_0, P_1$ respectively, at the conclusion of that execution of $\Psi$. 
• **(Correctness)** With overwhelming probability, \( k_0 = k_1 \) in an honest execution of \( \Psi \). That is, there is a negligible function \( \varepsilon \) such that for all \( \lambda \in \mathbb{N} \),

\[
\Pr_{(T, k_0, k_1) \leftarrow \Psi_\lambda} [k_0 = k_1] \geq 1 - \varepsilon(\lambda).
\]

• **(Secrecy)** The output key is indistinguishable from a random key, even in the presence of the corresponding protocol transcript. That is, for all \( \text{PPT} \ A \), there is a negligible function \( \varepsilon \) such that for all \( \lambda \in \mathbb{N} \),

\[
\Pr \left[ \begin{array}{l}
(T, k_0, k_1) \leftarrow \Psi_\lambda \\
b \leftarrow \{0, 1\} \\
k_0^* = k_0 \\
k_1^* \leftarrow \mathcal{K}_\lambda \\
b' \leftarrow A(1^\lambda, T, k_0^*)
\end{array} \right] \\
: b' = b \leq 1/2 + \varepsilon(\lambda).
\]

We define a pseudorandom key-exchange protocol to be a key-exchange protocol whose transcripts are distributed indistinguishably from random. That is, a pseudorandom key-exchange protocol has the stronger guarantee that \( (T, k_0, k_1) \leftarrow \Psi_\lambda \) is computationally indistinguishable from \( (T', k', k') \) where \( T' \) is a transcript comprising a sequence of uniformly random messages in the message space of \( \Psi_\lambda \), and \( k' \leftarrow \mathcal{K}_\lambda \).

The classical key-exchange protocol of Diffie and Hellman [63] is pseudorandom; in fact, its messages are uniformly random over a cyclic group \( G \). However, the constructions in this paper assume a key-exchange protocol whose messages are pseudorandom over **bit strings**. In fact, it is possible to transform a key-exchange protocol whose messages are pseudorandom over an arbitrary domain \( G \subseteq \{0, 1\}^\ell \) into a key-exchange protocol whose messages are pseudorandom over bit strings. Proposition 5.1.2, below, gives an encoding and decoding algorithm to transform uniformly random messages in \( G \) into a sequence of uniformly random messages in \( \{0, 1\}^\ell \). The encoding and decoding algorithms run in polynomial time as long as the density of messages \( |G|/2^\ell \) is noticeable (\( i.e., \) at least \( \frac{1}{\lambda^c} \) for some \( c \geq 1 \)). This is the case, for example, when the
Diffie-Hellman protocol is instantiated with the group of quadratic residues modulo a safe prime (in which case the density of message is constant close to \( \frac{1}{2} \)).

**Proposition 5.1.2.** Let \( G \) be a subset of \( \{0,1\}^\ell \) and define \( p = \frac{\lambda^\ell}{|G|} \). Consider the following encoding algorithm \( E : G \rightarrow (\{0,1\}^\ell)^p \) and decoding algorithm \( D : (\{0,1\}^\ell)^p \rightarrow G \cup \{\perp\} \) (\( \perp \) is a failure symbol):

- given input \( g \in G \), \( E \) performs the following steps:
  1. sample \( p \) independent and uniformly random \( \ell \)-bit strings: \( (s_1, \ldots, s_p) \).
  2. compute \( i = \min \{ j \in \{1, \ldots, p\} : s_j \in G \} \) (\( i = \infty \) if the minimum does not exist).
  3. if \( i \neq \infty \), change the value of \( s_i \) to \( g \).
  4. output \( (s_1, \ldots, s_p) \).

- given input \( (s_1, \ldots, s_p) \in (\{0,1\}^\ell)^p \), \( D \) performs the following steps:
  1. compute \( i = \min \{ j \in \{1, \ldots, p\} : s_j \in G \} \) (\( i = \infty \) if the minimum does not exist).
  2. if \( i \neq \infty \) output \( s_i \) otherwise output \( \perp \).

The algorithms \( E \) and \( D \) satisfy the following properties:

1. Correctness: for all \( g \in G \), \( \Pr[D(E(g)) \neq g] \) is negligible in \( \lambda \).
2. Randomness: for uniformly random \( g \leftarrow G \), \( E(g) \) is uniformly random over \( (\{0,1\}^\ell)^p \).

**Proof.** Correctness. Observe that \( D(E(g)) \neq g \) iff \( i = \infty \) in step 2 of \( E \). This occurs with probability \( \left(1 - \frac{|G|}{2^\ell}\right)^p \) by independence of \( (s_j)_{1 \leq j \leq p} \). For \( p = \frac{\lambda^\ell}{|G|} \), this probability is upper-bounded by \( \frac{1}{2^\lambda} \).

**Randomness.** Note that if \( s \) is uniform over \( \{0,1\}^\ell \), then conditioned on \( s \in G \), \( s \) is uniform over \( G \). Hence, for uniform input \( g \leftarrow G \), step 3 of \( E \) does not modify the distribution of \( s_i \).

**Remark 5.1.3.** An alternative encoding/decoding scheme similar to that of Proposition 5.1.2 could achieve better efficiency (namely, by encoding \( g \in G \) by substantially
fewer $\ell$-bit strings, in expectation), as outlined next. Our alternative encoding algorithm, instead of generating $p$ $\ell$-bit strings per group element $g$, generates random $\ell$-bit strings $s_1, s_2, \ldots$ until the first $i$ such that $s_i \in G$, and then outputs $s_1, \ldots, s_i$. This $i$ will usually be much smaller than $p$. The distribution of $s_1, \ldots, s_i$ is not uniformly random, however: it always ends with an element of $G$. To resolve this, we observe that what we really need is not that the encoding of each message in $G$ is a sequence of pseudorandom bit-strings, but rather that the encoding of a key-exchange transcript is a sequence of pseudorandom bit-strings. Hence, we can pad the entire transcript by appending additional random messages $s \in G$ until the transcript lies in $\left(\{0,1\}^\ell\right)^{p'}$ for some $p'$ chosen as follows: the probability that $p'$ is greater than the sum of lengths of the encodings of all the key-exchange messages must be overwhelming. In this scheme, $p'$ would be significantly smaller than $pL$, where $L$ is the number of messages in the key-exchange transcript.

5.1.2 Extractors

We will need the following definitions of two-source and seeded extractors.

**Definition 5.1.4** (Two-source extractors). The family $2\operatorname{Ext} : \{0,1\}^n \times \{0,1\}^{n'} \rightarrow \{0,1\}^\ell$ is a $(k_1, k_2, \varepsilon)$ two-source extractor if for all $\lambda \in \mathbb{N}$ and for all pairs $(X,Y)$ of independent random variables over $\{0,1\}^{n(\lambda)} \times \{0,1\}^{n'(\lambda)}$ such that $H_\infty(X) \geq k_1(\lambda)$ and $H_\infty(Y) \geq k_1(\lambda)$, it holds that:

$$\|2\operatorname{Ext}_\lambda(X,Y) - \mathcal{U}_{\ell(\lambda)}\|_s \leq \varepsilon(\lambda).$$

(5.1)

We say that $2\operatorname{Ext}$ is strong w.r.t. the first input if it satisfies the following stronger property:

$$\|(X, 2\operatorname{Ext}_\lambda(X,Y)) - (X, \mathcal{U}_{\ell(\lambda)})\|_s \leq \varepsilon(\lambda).$$

A strong two-source extractor w.r.t. the second input is defined analogously. Finally, we say that $2\operatorname{Ext}$ is a $(k, \varepsilon)$ same-source extractor if $n = n'$ and (5.1) is only required to hold when $(X,Y)$ is a pair of i.i.d. random variables with $H_\infty(X) = H_\infty(Y) \geq k(\lambda)$. 

131
Definition 5.1.5 (Seeded extractors). The family \( \text{Ext} : \{0,1\}^n \times \{0,1\}^{n'} \to \{0,1\}^\ell \) is a \((k, \varepsilon)\) seeded extractor if for all \( \lambda \in \mathbb{N} \) and any random variable \( X \) over \( \{0,1\}^{n(\lambda)} \) such that \( H_\infty(X) \geq k(\lambda) \), it holds that:

\[
\|\text{Ext}_\lambda(U_n(\lambda), X) - U_{\ell(\lambda)}\|_s \leq \varepsilon(\lambda).
\]

We say moreover that \( \text{Ext} \) is strong if it satisfies the following stronger property:

\[
\|\left(U_n(\lambda), \text{Ext}_\lambda(U_n(\lambda), X)\right) - (U_n(\lambda), U_{\ell(\lambda)})\|_s \leq \varepsilon(\lambda).
\]

5.1.3 Almost pairwise independent hashing

Definition 5.1.6 (\(\delta\)-almost pairwise independent hashing). A collection of functions \( H \) where each \( h \in H \) maps \( n \) bits to \( n' \) bits (i.e., \( h : \{0,1\}^n \to \{0,1\}^{n'} \)) is \(\delta\)-almost pairwise independent if

\[
\forall x \neq x' \in \{0,1\}^n \text{ and } y, y' \in \{0,1\}^{n'}, \Pr_{h \leftarrow H} [h(x) = y \land h(x') = y'] \leq \delta \cdot 2^{-n'}.
\]

5.2 Subliminal communication

Conversation Model. The protocols we will construct take place over a communication between two parties \( P_0 \) and \( P_1 \) alternatingly sending each other ciphertexts of a public-key encryption scheme. Without loss of generality, we assume that \( P_0 \) initiates the communication, and that communication occurs over a sequence of exchange-rounds each of which comprises two sequential messages: in each exchange-round, one party \( P_b \) sends a message to \( P_{1-b} \) and then \( P_{1-b} \) sends a message to \( P_b \). Let \( m_{b,i} \) denote the plaintext message sent by \( P_b \) to \( P_{1-b} \) in exchange-round \( i \), and let \( m_i = (m_{0,i}, m_{1,i}) \) denote the pair of messages exchanged. For \( i \geq 1 \), let us denote by

\[
\tau_{0,i} = (m_1, \ldots, m_{i-1}) \text{ and } \tau_{1,i} = (m_1, \ldots, m_{i-1}, m_{0,i})
\]
the plaintext transcripts available to $P_0$ and $P_1$ respectively during exchange-round $i$, in the case when $P_0$ sends the first message in exchange-round $i$.\footnote{If instead $P_1$ spoke first in round $i$, then $\tau_{0,i}$ would contain $m_{1,i}$, and $\tau_{1,i}$ would not contain $m_{0,i}$.} We define $\tau_{0,0}$ and $\tau_{1,0}$ to be empty lists (i.e., empty starting transcripts). (Note that when a notation contains both types of subscripts, we write the subscripts denoting the party and round in blue and red respectively, to improve readability.)

Recall that our adversary has the power to decrypt all ciphertexts under its chosen public-key encryption scheme $E$. Intuitively, it is therefore important that the plaintext conversation between $P_0$ and $P_1$ appears innocuous (and does not, for example, consist of ciphertexts of another encryption scheme). To model this, we assume the existence of a next-message distribution $\mathcal{M}$, which outputs a next innocuous message given the transcript of the plaintext conversation so far. This is denoted by $m_{b,i} \leftarrow \mathcal{M}(\tau_{b,i})$.

**Remark 5.2.1.** We emphasize that our main results make no assumptions at all on the distribution $\mathcal{M}$, and require only that the parties have oracle access to their own next-message distributions. Our main results hold in the presence of arbitrary message distributions: for example, they hold even in the seemingly inauspicious case when $\mathcal{M}$ is constant, meaning the parties are restricted to repeatedly exchanging a fixed message.

In Section 5.5, we discuss other more efficient constructions that can be used in settings where a stronger assumption — namely, that $\mathcal{M}$ has a certain amount of min-entropy — is acceptable. This stronger assumption, while not without loss of generality, might be rather benign in certain contexts (for example, if the messages exchanged are images).

In all the protocols we consider, the symbol $s$ is used to denote internal state kept locally by $P_0$ and $P_1$. It is implicitly assumed that each party’s state contains an up-to-date transcript of all messages received during the protocol. Parties may additionally keep other information in their internal state, as a function of the local
computations they perform. For \( i \geq 1 \), \( s_{b,i} \) denotes the state of \( P_b \) at the conclusion of exchange-round \( i \). Initial states \( s_{b,0} = \emptyset \) are empty.

We begin with a simpler definition that only syntactically allows for the transmission of a single message (Definition 5.2.2). This both serves as a warm-up to the multi-message definition presented next (Definition 5.2.4), and will be used in its own right to prove impossibility results. See Remark 5.2.7 for further discussion of the relationship between these two definitions.

**Definition 5.2.2.** A subliminal communication scheme is a two-party protocol:

\[
\Pi^E = (\Pi_{0,1}^E, \Pi_{1,1}^E, \Pi_{0,2}^E, \Pi_{1,2}^E, \ldots, \Pi_{0,r}^E, \Pi_{1,r}^E, \Pi_{1,\text{out}}^E)
\]

where \( r \in \text{poly} \) is the number of exchange-rounds and each \( \Pi_{b,i}^E \) is a PPT algorithm with oracle access to the algorithms of a public-key encryption scheme \( E \). Party \( P_0 \) is assumed to receive as input a message \( \mu \) (of at least one bit) that is to be conveyed to \( P_1 \) in an undetectable fashion. The algorithms \( \Pi_{b,i}^E \) are used by \( P_b \) in round \( i \), respectively, and \( \Pi_{1,\text{out}}^E \) denotes the algorithm run by \( P_1 \) to produce an output \( \mu' \) at the end of the protocol.

A subliminal communication scheme must satisfy the following syntax, correctness and security guarantees.

- **(Syntax)** In each exchange-round \( i = 1, \ldots, r \):
  
  - \( P_0 \) performs the following steps:
    
    1. Sample “innocuous message” \( m_{0,i} \leftarrow M(\tau_{0,i-1}) \).
    2. Generate ciphertext and state \( (c_{0,i}, s_{0,i}) \leftarrow \Pi_{0,i}^E(\mu, m_{0,i}, pk_1, s_{0,i-1}) \).
    3. Locally store \( s_{0,i} \) and send \( c_{0,i} \) to \( P_1 \).
  
  - Then, \( P_1 \) performs the following steps:\footnote{Note that the steps executed by \( P_0 \) and \( P_1 \) are entirely symmetric except in the following two aspects: first, \( P_0 \)’s input \( \mu \) is present in step 2 but not in step 2; and secondly, the state \( s_{1,i-1} \) used in step 2 contains the round-\( i \) message \( c_{0,i} \), whereas the state \( s_{0,i-1} \) used in step 2 depends only on the transcript until round \( i - 1 \).

    1. Sample “innocuous message” \( m_{1,i} \leftarrow M(\tau_{1,i-1}) \).}
2. Generate ciphertext and state \((c_{1,i}, s_{1,i}) \leftarrow \Pi^E_{1,i}(m_{1,i}, p_{k0}, s_{1,i-1})\).

3. Locally store \(s_{1,i}\) and send \(c_{1,i}\) to \(P_0\).

After \(r\) rounds, \(P_1\) computes \(\mu' = \Pi^E_{1,\text{out}}(s_{k1}, s_{1,r})\) and halts.

- **(Correctness)** For any \(\mu \in \{0, 1\}^\lambda\), if \(P_0\) and \(P_1\) play \(\Pi^E\) honestly, then \(\mu' = \mu\) with overwhelming probability. The probability is taken over the key generation \((p_{k0}, s_{k0}), (p_{k1}, s_{k1}) \leftarrow E.\text{Gen}\) and the randomness of the protocol algorithms, as well as the message distribution \(\mathcal{M}\).

- **(Subliminal indistinguishability)** For any semantically secure public-key encryption scheme \(E\), any \(\mu \in \{0, 1\}^\lambda\) and any next-message distribution \(\mathcal{M}\), for \((p_{k0}, s_{k0}), (p_{k1}, s_{k1}) \leftarrow E.\text{Gen}\), the following distributions are computationally indistinguishable:

<table>
<thead>
<tr>
<th>Ideal((p_{k0}, s_{k0}, p_{k1}, s_{k1}, \mathcal{M})):</th>
<th>Subliminal(<em>\Pi(\mu, p</em>{k0}, s_{k0}, p_{k1}, s_{k1}, \mathcal{M})):</th>
</tr>
</thead>
<tbody>
<tr>
<td>for (i = 1, \ldots, r):</td>
<td>for (i = 1, \ldots, r):</td>
</tr>
<tr>
<td>(m_{0,i} \leftarrow \mathcal{M}(\tau_{0,i}))</td>
<td>(m_{0,i} \leftarrow \mathcal{M}(\tau_{0,i}))</td>
</tr>
<tr>
<td>(m_{1,i} \leftarrow \mathcal{M}(\tau_{1,i}))</td>
<td>(m_{1,i} \leftarrow \mathcal{M}(\tau_{1,i}))</td>
</tr>
<tr>
<td>(c_{0,i} \leftarrow E.\text{Enc}(p_{k1}, m_{0,i}))</td>
<td>(c_{0,i} \leftarrow E.\text{Enc}(p_{k1}, m_{0,i}))</td>
</tr>
<tr>
<td>(c_{1,i} \leftarrow E.\text{Enc}(p_{k0}, m_{1,i}))</td>
<td>(c_{1,i} \leftarrow E.\text{Enc}(p_{k0}, m_{1,i}))</td>
</tr>
<tr>
<td>output</td>
<td>output</td>
</tr>
<tr>
<td>((p_{k0}, s_{k0}, p_{k1}, s_{k1}; (c_{0,i})_{b \in {0, 1}, i \in [r]}))</td>
<td>((p_{k0}, s_{k0}, p_{k1}, s_{k1}; (c_{0,i})_{b \in {0, 1}, i \in [r]}))</td>
</tr>
</tbody>
</table>

If the subliminal indistinguishability requirement is satisfied only for next-message distributions \(\mathcal{M}\) in a restricted set \(\mathcal{M}\), rather than for any \(\mathcal{M}\), then \(\Pi\) is said to be a subliminal communication scheme for \(\mathcal{M}\).

**Definition 5.2.3.** The rate of a subliminal communication protocol \(\Pi\) is defined as \(\frac{2r}{\lambda}\), where \(r\) is defined as in Definition 5.2.2. This is the average number of bits which are subliminally communicated per ciphertext of \(E\).

For simplicity, Definition 5.2.2 presents a communication scheme in which only a single hidden message \(\mu\) is transmitted. More generally, it is desirable to transmit
multiple messages, and bidirectionally, and perhaps in an adaptive manner.\(^8\) In multi-message schemes, it may be beneficial for efficiency that the protocol have a two-phase structure where some initial preprocessing is done in the first phase, and then the second phase can thereafter be invoked many times to transmit different hidden messages.\(^9\) This will a useful notion later in the paper, for our constructions, so we give the definition of a multi-message scheme here.

**Definition 5.2.4.** A multi-message subliminal communication scheme is a two-party protocol defined by a pair \((\Phi, \Xi)\) where \(\Phi\) ("Setup Phase") and \(\Xi\) ("Communication Phase") each define a two-party protocol. Each party outputs a state at the end of \(\Phi\), which it uses as an input in each subsequent invocation of \(\Xi\). An execution of a multi-message subliminal communication scheme consists of an execution of \(\Phi\) followed by one or more executions of \(\Xi\). More formally:

\[
\begin{align*}
\Phi^E & = (\Phi_{0,1}^E, \Phi_{1,1}^E, \Phi_{0,2}^E, \Phi_{1,2}^E, \ldots, \Phi_{0,r}^E, \Phi_{1,r}^E), \\
\Xi^E & = (\Xi_{0,1}^E, \Xi_{1,1}^E, \Xi_{0,2}^E, \Xi_{1,2}^E, \ldots, \Xi_{0,r'}^E, \Xi_{1,r'}^E, \Xi_{1,\text{out}}^E)
\end{align*}
\]

where \(r, r' \in \text{poly}\) are the number of exchange-rounds in \(\Phi\) and \(\Xi\) respectively, and where each \(\Phi_{b,i}^E, \Xi_{b,i}^E\) is a PPT algorithm with oracle access to the algorithms of a public-key encryption scheme \(E\). The protocol must satisfy the following syntax, correctness and security guarantees.

- **(Syntax)** In each exchange-round \(i = 1, \ldots, r\) of \(\Phi\): \(P_0\) executes the following steps for \(b = 0\), and then \(P_1\) executes the same steps for \(b = 1\).

1. Sample "innocuous message" \(m_{b,i} \leftarrow M(\tau_{b,i-1})\).

\(^8\)That is, the messages to be transmitted may become known as the protocol progresses, rather than all being known at the outset. This is the case, for example, if future messages depend on responses to previous ones.

\(^9\)As a concrete example: consider a simple protocol for transmitting a single encrypted message, consisting of key exchange followed by the transmission of message encrypted under the established key. When adapting this protocol to support multiple messages, it is beneficial to split the protocol into a one-time "phase 1" consisting of key exchange, and a "phase 2" encompassing the ciphertext transmission which can be invoked many times on different messages using the same phase-1 key. Such a protocol has much better amortized efficiency than simply repeating the single-message protocol many times, i.e., establishing a new key for each ciphertext.
2. Generate ciphertext and state \((c_{b,i}, s_{b,i}) \leftarrow \Phi^E_{b,i}(m_{b,i}, pk_{1-b, s_{b,i-1}})\).

3. Locally store \(s_{b,i}\) and send \(c_{b,i}\) to \(P_{1-b}\).

After the completion of \(\Phi\), either party may initiate \(\Xi\) by sending a first message of the \(\Xi\) protocol (with respect to a message \(\mu\) to be steganographically hidden, known to the initiating party). Let \(P_S\) and \(P_R\) denote the initiating and non-initiating parties in an execution of \(\Xi\), respectively.\(^\text{10}\) Let \(\mu \in \{0, 1\}^\lambda\) be the hidden message that \(P_S\) is to transmit to \(P_R\) in an undetectable fashion during an execution of \(\Xi\).

The execution of \(\Xi\) proceeds as follows over exchange-rounds \(i' = 1, \ldots, r'\):

- \(P_S\) acts as follows:
  1. Sample \(m_{S, r+i'} \leftarrow \mathcal{M}(\tau_{S, r+i'-1})\).
  2. Generate \((c_{S, r+i'}, s_{S, r+i'}) \leftarrow \Xi^E_{1,i'}(\mu, m_{S, r+i'}, pk_R, s_{S, r+i'-1})\).
  3. Locally store \(s_{S, r+i'}\) and send \(c_{S, r+i'}\) to \(P_R\).

- \(P_R\) acts as follows:
  1. Sample \(m_{R, r+i'} \leftarrow \mathcal{M}(\tau_{R, r+i'-1})\).
  2. Generate \((c_{R, r+i'}, s_{R, r+i'}) \leftarrow \Xi^E_{1,i'}(m_{R, r+i'}, pk_S, s_{R, r+i'-1})\).
  3. Locally store \(s_{R, r+i'}\) and send \(c_{R, r+i'}\) to \(P_S\).

At the end of an execution of \(\Xi\), \(P_R\) computes \(\mu' = \Xi^E_{1,\text{out}}(sk_1, s_{1, r+i'})\).

• (Correctness) For any \(\mu \in \{0, 1\}^\lambda\), if \(P_0\) and \(P_1\) execute \((\Phi, \Xi)\) honestly, then for every execution of \(\Xi\), the transmitted and received messages \(\mu\) and \(\mu'\) are equal with overwhelming probability. The probability is taken over the key generation \((pk_0, sk_0), (pk_1, sk_1) \leftarrow E.\text{Gen}\) and the randomness of the protocol algorithms, as well as the message distribution \(\mathcal{M}\).

• (Subliminal indistinguishability) For any semantically secure public-key encryption scheme \(E\), any polynomial \(p = p(\lambda)\), any sequence of hidden messages \(\bar{\mu} = (\mu_i)_{i \in [p]} \in (\{0, 1\}^\lambda)^p\), any sequence of bits \(\bar{b} = (b_1, \ldots, b_p) \in \{0, 1\}^p\) and any next-message distribution \(\mathcal{M}\), for \((pk_b, sk_b) \leftarrow E.\text{Gen}, b \in \{0, 1\}\) the following

\(^{10}\)Subscripts \(S, R \in \{0, 1\}\) stand for "sender" and "receiver," respectively.
distributions are computationally indistinguishable:

<table>
<thead>
<tr>
<th>Ideal((pk_0, sk_0, pk_1, sk_1, M)):</th>
<th>Subliminal(_{\Phi, \Xi}(\mu, \tilde{b}, pk_0, sk_0, pk_1, sk_1, M)):</th>
</tr>
</thead>
<tbody>
<tr>
<td>(for i = 1, \ldots, r + pr'):</td>
<td>(for i = 1, \ldots, r):</td>
</tr>
<tr>
<td>(m_{0,i} \leftarrow M(\tau_{0,i}))</td>
<td>(m_{0,i} \leftarrow M(\tau_{0,i}))</td>
</tr>
<tr>
<td>(m_{1,i} \leftarrow M(\tau_{1,i}))</td>
<td>(m_{1,i} \leftarrow M(\tau_{1,i}))</td>
</tr>
<tr>
<td>(c_{0,i} \leftarrow E.Enc(pk_1, m_{0,i}))</td>
<td>((c_{0,i}, s_{0,i}) \leftarrow \Phi_{0,i}^E(\mu, m_{0,i}, pk_1, s_{0,i-1}))</td>
</tr>
<tr>
<td>(c_{1,i} \leftarrow E.Enc(pk_0, m_{1,i}))</td>
<td>((c_{1,i}, s_{1,i}) \leftarrow \Phi_{1,i}^E(m_{1,i}, pk_0, s_{1,i-1}))</td>
</tr>
<tr>
<td>output: (\langle pk_0, sk_0, pk_1, sk_1; (c_{\beta,i})_{\beta \in {0,1}, i \in [r+pr']} \rangle)</td>
<td>output: (\langle pk_0, sk_0, pk_1, sk_1; (c_{\beta,i})_{\beta \in {0,1}, i \in [r+pr']} \rangle)</td>
</tr>
</tbody>
</table>

If the subliminal indistinguishability requirement is satisfied only for \(M\) in a restricted set \(\tilde{M}\), rather than for any \(M\), then \((\Phi, \Xi)\) is said to be a multi-message subliminal communication scheme for \(\tilde{M}\).

**Definition 5.2.5.** The asymptotic rate of a multi-message subliminal communication protocol \((\Phi, \Xi)\) is defined as \(\frac{r'}{r}\), where \(r'\) is defined as in Definition 5.2.4. The asymptotic rate is the average number of bits which are subliminally communicated per ciphertext exchanged between \(P_0\) and \(P_1\) after the one-time setup phase is completed.

**Definition 5.2.6.** The setup cost of a multi-message subliminal communication protocol \((\Phi, \Xi)\) is defined as \(r\), i.e., the number of rounds in \(\Phi\). The setup cost is the number of ciphertexts which must be sent back and forth between \(P_0\) and \(P_1\) in order to complete the setup phase.
Remark 5.2.7. Definition 5.2.4 (multi-message subliminal communication scheme) is equivalent to Definition 5.2.2 (subliminal communication scheme) in the sense that the existence of any single-message scheme trivially implies a multi-message scheme and vice versa. We present Definition 5.2.4 as it will be useful for presenting and analyzing asymptotic efficiency of our constructions, but note that this equivalence means that the simpler Definition 5.2.2 suffices in the context of impossibility (or possibility) results, such as that given in Section 5.3.

5.3 Impossibility results

5.3.1 Locally decodable subliminal communication schemes

A first attempt at achieving subliminal communication might consider schemes with the following natural property: the receiving party $P_1$ extracts hidden bits one ciphertext at a time, by the application of a single (possibly randomized) decoding function. We refer to such schemes as locally decodable and our next impossibility theorem shows that non-trivial locally decodable schemes do not exist if the encryption scheme $E$ is chosen adversarially.

Theorem 5.3.1. For any locally decodable protocol $\Pi$ satisfying the syntax of a single-message\textsuperscript{11} subliminal communication scheme (Definition 5.2.2), there exists a semantically secure public-key encryption scheme $E$ such that $E$ violates the correctness condition of Definition 5.2.2. Therefore, no locally decodable protocol $\Pi$ is a subliminal communication scheme.

Proof. Let us consider a locally decodable scheme such as in the statement of the theorem, and let us denote by $\Pi_{1,\text{out}} : \{0, 1\}^n \rightarrow \{0, 1\}$ the decoding function of the scheme where the input is a ciphertext $c$. We now construct an encryption scheme which biases the output of $\Pi_{1,\text{out}}$ arbitrarily close to a constant bit. This is a contradiction, since by correctness and subliminal indistinguishability, $\Pi_{1,\text{out}}(c)$ should have negligible bias when subliminally communicating a uniformly random message.

\textsuperscript{11}Remark 5.2.7 discusses the sufficiency of proving impossibility for single-message schemes.
\( \mu \leftarrow \{0, 1\} \) (without loss of generality, we assume the subliminal communication scheme transmits single-bit messages).

Let \( E \) be any semantically secure encryption scheme with ciphertext space \( C = \{0, 1\}^n \) and message space \( M \). Without loss of generality we assume that for at least half the messages \( m \in M \), we have \( \Pr[\Pi_{1,\text{out}}(E.\text{Enc}(pk, m)) = 1] \geq \frac{1}{2} \) (otherwise we can just replace 1 by 0 in the construction below). We now define the encryption scheme \( E' \) which is identical to \( E \) except for \( E'.\text{Enc} \) which on input \((pk, m)\) runs as follows for some constant \( t \).

1. Repeat at most \( t \) times:
   (a) Sample encryption \( c \leftarrow E.\text{Enc}(pk, m) \).
   (b) If \( \Pi_{1,\text{out}}(c) = 1 \), exit the loop; otherwise, continue.

2. Output \( c \).

It is clear that \( E' \) is also semantically secure: oracle access to \( E'.\text{Enc} \) can be simulated with oracle access to \( E.\text{Enc} \), so a distinguisher which breaks the semantic security of \( E' \) can also be used to break the semantic security of \( E \). Finally, for a message \( m \) such that \( \Pr[\Pi_{1,\text{out}}(E.\text{Enc}(pk, m)) = 1] \geq \frac{1}{2} \), by definition of \( E'.\text{Enc} \), it holds that

\[
\Pr[\Pi_{1,\text{out}}(E'.\text{Enc}(PK, m)) = 1] \geq 1 - \frac{1}{2^t}.
\]

This shows that the output of \( \Pi_{1,\text{out}} \) can be arbitrarily biased and concludes the proof.

\( \square \)

**Remark 5.3.2.** The essence of the above theorem is the impossibility of deterministic extraction: no single deterministic function can deterministically extract from ciphertexts of arbitrary encryption schemes. The way to bypass this impossibility is to have the extractor depend on the encryption scheme. Note that multiple-source extraction, which is used in our constructions in the subsequent sections, implicitly do depend on the underlying encryption scheme, since the additional sources of input depend on the encryption scheme and thus can be thought of as “auxiliary input” that is specific to the encryption scheme at hand.
5.3.2 Steganography for adversarial cover distributions

Our second impossibility result concerns a much more general class of communication schemes, which we call *steganographic communication schemes*. Subliminal communication schemes, as well as the existing notions of public-key steganography and steganographic key exchange from the steganography literature, are instantiations of the more general definition of a (multi-message) steganographic communication scheme. To our knowledge, the general notion of a steganographic communication scheme has not been formalized in this way in prior work. In the context of this work, the general definition is helpful for proving broad impossibilities across multiple types of steganographic schemes.

As mentioned in the introduction, a limitation of all existing results in the steganographic literature, to our knowledge, is that they assume that the *cover distribution* — i.e., the distribution of innocuous objects in which steganographic communication is to be embedded — is fixed *a priori*. In particular, the cover distribution is assumed not to depend on the description of the steganographic communication scheme. The impossibility result given in Section 5.3.1 is an example illustrative of the power of adversarially choosing the cover distribution: Theorem 5.3.1 says that by choosing the encryption scheme $E$ to depend on a given subliminal communication scheme, an adversary can rule out the possibility of any hidden communication at all.

Our next impossibility result (Theorem 5.3.3) shows that if the cover distribution is chosen adversarially, then non-trivial steganographic communication is impossible.

**Theorem 5.3.3.** Let $\Pi$ be a protocol with the syntax of a steganographic communication scheme. Then for any $k \in \mathbb{N}$, there exists a cover distribution $C$ of conditional min-entropy $k$ such that steganographic indistinguishability of $\Pi$ does not hold.

In Section 5.6, we give the formal definition of a *steganographic communication scheme*, along with the proof of Theorem 5.3.3. We have elected to present these in a later section as the definition introduces a set of somewhat involved new notation only used for the corresponding impossibility result. Both the definition and the impossibility result are somewhat tangential to the main results of this work, which
are presented in the immediately following sections and whose focus is on subliminal communication schemes.

5.4 Construction of the subliminal scheme

The goal of this section is to establish the following theorem, which states that our construction \((\Phi^*, \Xi^*)\) is a subliminal communication scheme when instantiated with a pseudorandom key-exchange protocol (such as Diffie-Hellman).

**Theorem 5.4.1.** The protocol \((\Phi^*, \Xi^*)\) given in Definition 5.4.15, when instantiated with a pseudorandom key-exchange protocol \(\Psi\), is a multi-message subliminal communication scheme.

The detailed description and proofs of security and correctness of our scheme can be found in the following subsections. Our construction makes no assumption on the message distribution \(M\) and in particular holds when the exchanged plaintexts (of the adversarially mandated encryption scheme \(E\)) are a fixed, adversarially chosen sequence of messages. An informal outline of the construction is given next.

**Definition 5.4.2. Outline of the construction.**

1. **Setup Phase \(\Phi^*\)**

   (a) A \(\tilde{O}(\log \lambda)\)-bit string \(S\) is established between \(P_0\) and \(P_1\) by extracting randomness from pairs of consecutive ciphertexts. (Protocol overview in Section 5.4.1.)

   (b) Let \(\text{Ext}\) be a strong seeded extractor, and let \(S\) serve as its seed. By rejection-sampling ciphertexts \(c\) until \(\text{Ext}_S(c) = \chi\), either party can embed a random string \(\chi\) of their choice in the conversation. (Protocol overview in Section 5.4.2.) By embedding in this manner the messages of a pseudorandom key-exchange protocol, both parties establish a shared secret \(sk^*.\)\(^{12}\)

---

\(^{12}\)Note that the random string \(\chi\) is known to an eavesdropper who has knowledge of the seed \(S\). Nonetheless, (1) the established secret \(sk^*\) is unknown to the eavesdropper by the security of the key-exchange protocol and (2) the transcript is indistinguishable to the eavesdropper from one in which no key exchange occurred at all, due to the pseudorandomness of the key-exchange messages.
2. Communication Phase $\Xi^*$

Both parties can now communicate arbitrary messages of their choice by (1) encrypting them using a pseudorandom secret-key encryption scheme $\text{SKE}$ using $sk^*$ as the secret key, and (2) embedding the ciphertexts of $\text{SKE}$ using the rejection-sampling technique described in Step 1b.$^{13}$ (Detailed protocol in Section 5.4.3.)

The full protocol is given, and proven to be a subliminal communication scheme, in Section 5.4.3.

5.4.1 Establishing a shared seed

In this section, we give a protocol which allows $P_0$ and $P_1$ to establish a random public parameter which will be used in subsequent phases of our subliminal scheme. As such, this can be thought of as drawing a subliminal scheme at random from a family of subliminal schemes. The parameter is public in the sense that anyone eavesdropping on the channel between $P_0$ and $P_1$ gains knowledge of it. A crucial point is that the random draw occurs after the adversarial encryption scheme $E$ is fixed, thus bypassing the impossibility results of Section 5.3.

Our strategy is simple: extract randomness from pairs of ciphertexts. Since the extractor does not receive the key, semantic security holds with respect to the extractor: a pair of ciphertexts for two arbitrary messages is indistinguishable from two encryptions of a fixed message; thus, a same-source extractor suffices for our purposes (see Lemma 5.4.6). Even though semantic security guarantees only $\omega(\log \lambda)$ min-entropy of ciphertexts (see Lemma 5.4.4), we will be able to make use of the "greater-than" extractor (Definition 5.4.3) applied to pairs of ciphertexts, and obtain Theorem 5.4.8.

**Definition 5.4.3.** The greater-than extractor $\text{GT}$ is defined by $\text{GT}(x, y) \triangleq 1[x \geq y]$.

**Lemma 5.4.4** (Ciphertexts have super-logarithmic min-entropy). Let $E$ be a semantically secure encryption scheme. Then there exists a negligible function $\varepsilon$ such that

$^{13}$Again, an eavesdropper could know the SKE ciphertexts exchanged, if he knew the seed $S$, but could not distinguish the SKE ciphertexts from truly random strings, and thus could not tell whether any subliminal communication was occurring at all. Cf. footnote 12.
for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}_\lambda$, writing $C_{m}^{pk} \sim \text{E.Enc}(pk, m)$:

$$\Pr \left[ (pk, sk) \leftarrow \text{E.Gen}(1^\lambda) : H_\infty(C_{m}^{pk}) \geq \log \frac{1}{\varepsilon(\lambda)} \right] \geq 1 - \varepsilon(\lambda).$$

Proof. Let us assume for contradiction that there exists $p \in \text{poly}(\lambda)$, a countably infinite set $I$ and a family $\{m_\lambda\}_{\lambda \in I}$ such that for all $\lambda \in I$: 

$$\Pr \left[ (pk, sk) \leftarrow \text{E.Gen}(1^\lambda) : H_\infty(C_{m_\lambda}^{pk}) \leq \log p(\lambda) \right] \geq \frac{1}{p(\lambda)}. \quad (5.2)$$

Since $E$ is non-trivial, for all $\lambda \in I$ there exists $m_\lambda' \neq m_\lambda$.

Consider the distinguisher $D$ which on input $(pk, c)$ runs as follows:

1. Sample encryption $e \leftarrow \text{E.Enc}(pk, m_\lambda)$.
2. If $e = c$ output 1, otherwise output 0.

Writing $(PK, SK) \sim \text{E.Gen}(1^\lambda)$, we claim that $D$ distinguishes $(PK, \text{Enc}(PK, m_\lambda))$ from $(PK, \text{Enc}(PK, m_\lambda'))$ with non-negligible advantage. First note that by (5.2), with probability at least $\frac{1}{p(\lambda)}$ over the draw of $pk$, $\text{Enc}(pk, m_\lambda)$ has collision probability at least $\frac{1}{p(\lambda)^2}$. By definition, $D$ outputs 1 on input $(pk, \text{Enc}(pk, m_\lambda))$ if and only if a collision occurs in step 1. Hence:

$$\Pr[D(PK, \text{Enc}(PK, m_\lambda)) = 1] \geq \frac{1}{p(\lambda)^3}.$$}

Second, since $E$ is correct, there exists a negligible function $\varepsilon'$ such that $\text{Enc}(pk, m_\lambda) = \text{Enc}(pk, m_\lambda')$ with probability at most $\varepsilon'$. Hence:

$$\Pr[D(PK, \text{Enc}(PK, m_\lambda')) = 1] \leq \varepsilon'(\lambda).$$

The previous two inequalities together imply that for large enough $\lambda \in I$:

$$\left| \Pr[D(PK, \text{Enc}(PK, m_\lambda)) = 1] - \Pr[D(PK, \text{Enc}(PK, m_\lambda')) = 1] \right| \geq \frac{1}{2p(\lambda)^3},$$

which contradicts the semantic security of $E$. $\square$
Given that ciphertexts of semantically secure encryption schemes have min-entropy $\omega(\log \lambda)$, we will consider extractors which have negligible bias on such sources. This motivates the following definition.

**Definition 5.4.5.** Let $2\text{Ext} : \{0, 1\}^n \times \{0, 1\}^m \to \{0, 1\}^\ell$ be a two-source extractor, we say that $2\text{Ext}$ is an extractor for super-logarithmic min-entropy if $2\text{Ext}$ is a $(d \log \lambda, d \log \lambda, \frac{1}{\lambda^d})$ extractor for any $d \in \mathbb{N}$. In particular, for any negligible function $\epsilon$, there exists a negligible function $\epsilon'$ such that $2\text{Ext}$ is a $(\log \frac{1}{\epsilon}, \log \frac{1}{\epsilon}, \epsilon')$ extractor.

The following lemma shows that the output of a same-source extractor for super-logarithmic min-entropy on two ciphertexts is statistically indistinguishable from uniform, even in the presence of the key.

**Lemma 5.4.6.** Let $E$ be a semantically secure encryption scheme with ciphertext length $n$, and let $2\text{Ext} : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}^\ell$ be a same-source extractor for super-logarithmic min-entropy with $\ell(\lambda) = O(\log \lambda)$, then there exists a negligible function $\epsilon$ such that, for any $\lambda \in \mathbb{N}$, $(m_0, m_1) \in M_\lambda^2$, writing $(PK, SK) \sim E.\text{Gen}(1^\lambda)$, $C_i^{pk} \sim E.\text{Enc}(pk, m_i)$, $i \in \{0, 1\}$:

$$\| (PK, SK, 2\text{Ext}(C_0^{PK}, C_1^{PK})) - (PK, SK, \mathcal{U}_{\ell(\lambda)}) \|_S \leq \epsilon(\lambda).$$

**Proof.** We will prove that for any polynomial $p$ and for large enough $\lambda$:

$$\Pr \left[ (pk, sk) \leftarrow E.\text{Gen}(1^\lambda) : \| 2\text{Ext}(C_0^{pk}, C_1^{pk}) - \mathcal{U}_{\ell(\lambda)} \|_S \leq \frac{1}{p(\lambda)} \right] \geq 1 - \frac{1}{p(\lambda)}.$$

Assume by contradiction that there exists $p \in \text{poly}(\lambda)$ and an infinite set $I \subseteq \mathbb{N}$ such that for $\lambda \in I$:

$$\Pr \left[ (pk, sk) \leftarrow E.\text{Gen}(1^\lambda) : \| 2\text{Ext}(C_0^{pk}, C_1^{pk}) - \mathcal{U}_{\ell(\lambda)} \|_S \geq \frac{1}{p(\lambda)} \right] \geq \frac{1}{p(\lambda)}. \quad (5.3)$$

We now construct an adversary $D$ distinguishing between $(PK, C_0^{PK})$ and $(PK, C_1^{PK})$ with non-negligible advantage. On input $(pk, c)$, $D$ runs as follows:

1. Sample two encryptions of $m_0$: $c_0, c'_0 \leftarrow E.\text{Enc}(pk, m_0)$, and $c_1 \leftarrow E.\text{Enc}(pk, m_1)$.
2. If $2\text{Ext}(c_0, c_1) = 2\text{Ext}(c'_0, c)$ output 1, otherwise output 0.

First, note that on input $(PK, C^{PK}_1)$, $D$ outputs 1 iff a collision occurs at step 2. By (5.3), with probability at least $\frac{1}{p}$ over the draw of the $(pk, sk)$, a collision occurs with probability at least $\frac{1}{2^{2f}} + \frac{1}{2^{f^2}}$. Otherwise, a collision occurs with probability at least $\frac{1}{2^{f}}$. Overall, for $\lambda \in I$:

$$\Pr[D(PK, C^{PK}_1) = 1] \geq \frac{1}{2^{f}} + \frac{1}{2^{f^2}}.$$  \hspace{1cm} (5.4)

By Lemma 5.4.4, after conditioning on the event that $H_{\infty}(C^{pk}_0) \geq \log q(\lambda)$, the guarantee of $2\text{Ext}$ applies to a pair of independent encryptions of $m_0$ under $pk$ and we obtain, for large enough $\lambda$:

$$\Pr \left[ (pk, sk) \leftarrow \text{E.Gen}(1^\lambda) : \left\| 2\text{Ext}(C^{pk}_0, C^{pk}_0) - \mathcal{U}_0(\lambda) \right\|_s \leq \frac{1}{q(\lambda)} \right] \geq 1 - \frac{1}{q(\lambda)},$$

This implies that for large enough $\lambda$:

$$\Pr[D(PK, C^{PK}_0) = 1] \leq \frac{1}{2^{f^2}} + \frac{2}{q}. \hspace{1cm} (5.5)$$

Together, (5.4) and (5.5) imply, after choosing $q = 2^{f^2 + 2p^3} \in \text{poly}$, that for large enough $\lambda \in I$:

$$\Pr[D(PK, C^{PK}_0) = 1] - \Pr[D(PK, C^{PK}_1) = 1] \geq \frac{2}{q}.$$  

This contradicts the security of $E$ and concludes the proof. \hfill \Box

Finally, we observe that the "greater-than" extractor is a same-source extractor for super-logarithmic min-entropy (Lemma 5.4.7). To the best of our knowledge, this is a folklore fact which is for example mentioned in [17].

**Lemma 5.4.7.** For any $k \leq n$, $\text{GT}$ is a $(k, \frac{1}{2^k})$ same-source extractor.
Proof. Let $X$ and $Y$ be two i.i.d. variables with $H_\infty(X) = H_\infty(Y) \geq k$. Then

$$2 \Pr[X \geq Y] = \Pr[X \geq Y] + \Pr[X \leq Y] = 1 + \mathbb{C}_p(X) \leq 1 + \frac{1}{2^k},$$

where the last inequality uses the standard upper bound on the collision probability $\mathbb{C}_p(X)$ with respect to the min-entropy of $X$. \hfill \square

We now conclude this section with a full description of our method for establishing the public parameter $S$ introduced in Step 1a.

**Theorem 5.4.8.** Let $E$ be a semantically secure public-key encryption scheme and let $\rho \in \text{poly}$. Define random variables as follows.

- For $b \in \{0, 1\}$, let $K_b = (PK_b, SK_b) = E.Gen(1^\lambda)$.
- For $b \in \{0, 1\}$ and $i \in [2\rho]$, let $C_{b,i} = E.Enc(PK_{1-b}, m_{b,i})$ representing the ciphertexts exchanged between $P_0$ and $P_1$ during $2\rho$ exchange-rounds.
- Let $S = (\text{GT}(C_{0,1}, C_{0,2}), \text{GT}(C_{1,1}, C_{1,2}), \ldots, \text{GT}(C_{0,2\rho-1}, C_{0,2\rho}), \text{GT}(C_{1,2\rho-1}, C_{1,2\rho}))$.

There exists a negligible function $\varepsilon$ such that:

$$\|(K_0, K_1, S) - (K_0, K_1, U_{2\rho})\| \leq \varepsilon.$$

Proof. Writing $S = (S_1, S_1', \ldots, S_\rho, S_\rho')$, we have:

$$\|(K_0, K_1, S) - (K_0, K_1, U_{2\rho})\| \leq \sum_{i=1}^\rho (\|(K_0, K_{1,i}) - (K_0, K_1, U_1)\| + \|(K_0, K_1, S_i') - (K_0, K_1, S_1)\|) \leq 2\rho\varepsilon,$$

where the first inequality follows by independence of the ciphertexts conditioned on the keys, and the second inequality follows by Lemma 5.4.6. \hfill \square

**Remark 5.4.9.** In the construction of Theorem 5.4.8, the ciphertexts exchanged between $P_0$ and $P_1$ are sent without any modification, so subliminal indistinguishability clearly holds at this point.
5.4.2 Embedding random strings

In this section, we assume that both parties have access to a public parameter $S$ and construct a protocol which allows for embedding of uniformly random strings into ciphertexts of an adversarially chosen encryption scheme $E$, as required by Steps 1b and 2 of the construction outline (Definition 5.4.2). The security guarantee is that for a uniformly random parameter $S$ and uniformly random strings to be embedded, the ciphertexts of $E$ with embedded random strings are indistinguishable from ciphertexts of $E$ produced by direct application of $E$.Enc, even to an adversary who knows the decryption keys of $E$. This can be thought of as a relaxation of subliminal indistinguishability (Definition 5.2.2) where the two main differences are that (1) the parties have shared knowledge of a random seed, and (2) indistinguishability only holds when embedding a random string, rather than for arbitrary strings. We first present a construction to embed logarithmically many random bits (Theorem 5.4.10) and then show how to sequentially compose it to embed arbitrarily polynomially many random bits (Theorem 5.4.12). These constructions rely on a strong seeded extractor that can extract logarithmically many bits from sources of super-logarithmic min-entropy. Almost universal hashing is a simple such extractor, as stated in Proposition 5.4.13.

**Theorem 5.4.10.** Let $\text{Ext} : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^u$ be a strong seeded extractor for super-logarithmic min-entropy with $u = O(\log \lambda)$, and let $E$ be a semantically secure encryption scheme with ciphertext space $C = \{0,1\}^n$. Let $\Sigma^{E,S}$ be defined as in Algorithm 10, then the following guarantees hold:

<table>
<thead>
<tr>
<th>Algorithm 10 Rejection sampler $\Sigma^{E,S}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PUBLIC PARAMETER:</strong> $S$ (a $d$-bit seed).</td>
</tr>
<tr>
<td><strong>INPUT:</strong> $(\chi, m, pk)$ where $\chi$ is the string to be embedded.</td>
</tr>
<tr>
<td>1. Generate encryption $c \leftarrow E.\text{Enc}(pk, m)$.</td>
</tr>
<tr>
<td>2. If $\text{Ext}(r, c) = \chi$, then output $c$. Else, go back to step 1.</td>
</tr>
</tbody>
</table>

1. Correctness: for any $S \in \{0,1\}^d$ and $\chi \in \{0,1\}^u$, if $c = \Sigma^{E,S}(\chi, m, pk)$, and $\chi' = \text{Ext}(S, c)$, then $\chi' = \chi$.  

148
2. Security: there exists a negligible function \( \varepsilon \) such that writing \( (PK, SK) = E.Gen(1^\lambda), C = E.Enc(PK, m) \) and \( C' = \Sigma^{E,U}(U_v, m, PK) \), the following holds:

\[
\| (PK, SK, U_d, C) - (PK, SK, U_d, C') \|_s \leq \varepsilon(\lambda).
\]

Proof. Define \( C'' = E.Enc(PK_2, m) \), an encryption of \( m \) independent of \( C \). By definition of rejection sampling, \( C \sim \Sigma^{E,S}(Ext(S, C''), m, PK_2) \). Since \( Ext \) is a strong extractor for super-logarithmic min-entropy, and since \( C \) has super-logarithmic min-entropy, there exists a negligible function \( \varepsilon \) such that:

\[
\| (PK_2, SK_2, U_d, Ext(U_d, C'')) - (PK_2, SK_2, U_d, U_v) \|_s \leq \varepsilon(\lambda).
\]

The statistical distance can only decrease by applying \( \Sigma^E \) on both sides, hence:

\[
\| (PK_2, SK_2, U_d, C) - (PK_2, SK_2, U_d, C') \|_s \leq \varepsilon(\lambda).
\]

which proves the security guarantee. Correctness is immediate. \( \square \)

Remark 5.4.11. Rejection sampling is a simple and natural approach that has been used by prior work in the steganographic literature, such as [15]. Despite the shared use of this common technique, our construction is more different from prior art than it might seem at first glance. The novelty of our construction arises from the challenges of working in a model with a stronger adversary who can choose the distribution of ciphertexts (i.e., the adversary gets to choose the public-key encryption scheme \( E \)). We manage to bypass the impossibilities outlined in Section 5.3 notwithstanding this stronger adversarial model, and in contrast to prior work, construct a protocol to established a shared seed from scratch, rather than simply assuming that one has been established in advance.

We now sequentially compose Theorem 5.4.10 to embed longer strings.

Theorem 5.4.12. Let \( \Sigma \) be the rejection sampler defined in Algorithm 10. Let \( \ell \in \text{poly} \) and \( U_\ell \) be a uniformly random message of \( \ell \) bits. For \( v \leq \ell \), we write
\( U_t = U_{t,1} \ldots U_{t,v} \) where \( U_{t,i} \) is a block of \( v \) bits from \( U_t \) and \( v = \frac{t}{v'} \). Given cover messages \( m_1, \ldots, m_t \), define \((PK, SK) = E.Gen(1^\lambda)\), \( C_i' = \sum E_{U_{t,i}}(U_{t,i}, m_i, pk) \), \( C_i = E.Enc(PK, m_i) \), then there exists a negigible function \( \varepsilon \) such that:

\[
\|(PK, SK, U_d, (C_i)_{i \in [v]}) - (PK, SK, U_d, (C_i')_{i \in [v]})\|_s \leq \varepsilon(\lambda)
\]

Proof. Define \( K = (PK, SK) \), then:

\[
\|(K, U_d, (C_i)_{i \in [v]}) - (K, U_d, (C_i')_{i \in [v]})\|_s \leq \sum_{i=1}^{v} \|(K, C_i) - (K, C_i')\|_s \leq v \varepsilon,
\]

where the first inequality is by independence of the sequences \((C_i)_{i \in [v]}\) and \((C_i')_{i \in [v]}\) conditioned on the keys, and the second inequality is by Theorem 5.4.10.

Finally, we observe that almost universal hashing is a strong seeded extractor for super-logarithmic min-entropy which has negligible error when the output length is \( O(\log(\lambda)) \) (Proposition 5.4.13). This exactly satisfies the requirement of Theorem 5.4.10. Moreover, the seed length of this extractor is only super-logarithmic, meaning that the seed can be established, in Step 1a of the Setup Phase (Definition 5.4.2), in \( \tilde{O}(\log \lambda) \) many exchange-rounds of communication.

**Proposition 5.4.13.** Let \( \delta \) be a negligible function and let \( H \) be a set of \( \delta \)-almost pairwise independent hash functions mapping \( \{0, 1\}^n \) to \( \{0, 1\}^{c \log n} \), then the extractor \( Ext : H \times \{0, 1\}^n \rightarrow \{0, 1\}^{c \log n} \) defined by \( Ext(h, x) = h(x) \) is a strong seeded extractor for super-logarithmic min-entropy. Furthermore, there exists an explicit set \( H \) of such hash functions, such that sampling uniformly from \( H \) requires \( O(c \cdot \log n + \log \frac{1}{\delta}) \) bits.

Proof. See [168].

### 5.4.3 Full protocol \((\Phi^*, \Xi^*)\)

First, we establish some notation for the syntax of a key-exchange protocol.

**Definition 5.4.14** (Key-exchange protocol syntax). A key-exchange protocol is a
two-party protocol defined by

$$\Psi = (\Psi_{0,1}, \Psi_{1,1}, \Psi_{0,2}, \Psi_{1,2}, \ldots, \Psi_{0,k}, \Psi_{1,k}, \Psi_{0,\text{out}}, \Psi_{1,\text{out}})$$

We assume $k$ simultaneous communication rounds, where $\Psi_{b,i}$ represents the computation performed by $P_b$ in the $i$th round. The parties are stateful and their state is implicitly updated at each round to contain the transcript so far and any local randomness generated so far. Each $\Psi_{b,i}$ takes as input the transcript up to round $i - 1$ and the state of $P_b$, and outputs a message $\psi_{b,i}$ to be sent in the $i$th round. For notational simplicity, we write explicitly only the first input to $\Psi_{b,i}$, and leave the second input (i.e., the state) implicit. $\Psi_{0,\text{out}}, \Psi_{1,\text{out}}$ are run by $P_0, P_1$ respectively to compute the shared secret at the conclusion of the protocol.

Next, we give the full construction of $(\Phi^*, \Xi^*)$ following the outline in Definition 5.4.2.

Definition 5.4.15. $(\Phi^*, \Xi^*)$ is parametrized by the following.

- $\text{Ext} : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^v$, a strong seeded extractor.
- $\Psi$, a pseudorandom key-exchange protocol with $\ell$-bit messages.
- $\text{SKE}$, a pseudorandom secret key encryption scheme with $\xi$-bit ciphertexts.

We define each phase of our construction in turn.

1. **Setup Phase $\Phi^*$**
   
   (a) **Establishing a $d$-bit shared seed**

   - For $b \in \{0,1\}$ and $i \in \{1, \ldots, d\}$, $\Phi^*_{b,i}(m_{b,i}, pk_{1-b}, s_{b,i-1})$ outputs a ciphertext $c_{b,i} = E.\text{Enc}(pk_{1-b}, m_{b,i})$ and sets the updated state $s_{b,i}$ to be the transcript of all protocol messages sent and received so far.
   - At the conclusion of the $d$ exchange-rounds, each party updates his state

---

14In presenting our construction $(\Phi^*, \Xi^*)$, we do not denote the state of parties w.r.t. the key-exchange protocol $\Psi$ by a separate variable, but assume that it is part of the state $s_{b,i}$ of the overall protocol.
to contain the seed $S$ which is defined by

$$S = (\text{GT}(c_{0,1}, c_{0,2}), \text{GT}(c_{1,1}, c_{1,2}), \ldots, \text{GT}(c_{0,d-1}, c_{0,d}), \text{GT}(c_{1,d-1}, c_{1,d})).$$

This seed $S$ is assumed to be accessible in all future states throughout both phases during the remainder of the protocol.

(b) **Subliminal key exchange**

Let $\nu \triangleq \frac{\xi}{\nu}$. Subliminal key exchange occurs over $k \cdot \nu$ exchange-rounds.

- For $j \in \{1, \ldots, k\}$ and $b \in \{0, 1\}$:
  - $P_b$ retrieves from his state the key-exchange transcript so far $(\psi_{b,j'})_{b \in \{0,1\}, j'<j}$.
  - $P_b$ computes the next key-exchange message

$$\psi_{b,j} \leftarrow \Psi_{b,j}((\psi_{b,j'})_{b \in \{0,1\}, j'<j}).$$

- $P_b$ breaks $\psi_{b,j}$ into $\nu$-bit blocks $\psi_{b,j} = \psi_{b,j}^1 \parallel \ldots \parallel \psi_{b,j}^\nu$.
- The $\nu$ blocks are transmitted sequentially as follows. For $i \in \{1, \ldots, \nu\}$:

  Let $i = d + (j - 1)\nu + \iota$.
  $\Phi^*_{b,i}(m_{b,i}, pk_{1-b, s_{b,i-1}})$ outputs $c_{b,i} \leftarrow \Sigma^E_{\psi_{b,j}^i}(m_{b,i}, pk_{1-b})$ and sets the updated state $s_{b,i}$ to contain the transcript of all protocol messages sent and received so far.

- At the conclusion of the $\nu$ exchange-rounds, each party updates his state to contain the secret key $sk^*$ computed as follows:

$$sk^* = \text{SKE}.\text{SGen} \left(1^\Lambda; \Psi_{\text{out}}((\psi_{b,j})_{b \in \{0,1\}, j \in [k]}) \right).$$
2. Communication Phase $\Xi^*$

Each communication phase occurs over $r' \overset{\Delta}{=} \xi/v$ exchange-rounds.

Let $\beta \in \{0, 1\}$ be the initiating party and let $\bar{\beta} = 1 - \beta$.

$P_{\beta}$ performs the following steps.

- Generate $c^* \leftarrow \text{SKE.SEnc}(sk^*, \mu)$.
- Break $c^*$ into $v$-bit blocks $c^* = c^*_1 || \ldots || c^*_v$.

For $i' \in \{1, \ldots, r'\}$:

- Let $i'' = r + i'$.
- $\Xi^*_{0,i''}(\mu, m_{0,i''}, pk_{\bar{\beta}}, s_{\beta,i''-1})$ outputs $c_{\beta,i''} \leftarrow \Sigma^{E,S}(c^*_{i'}, m_{\beta,i''}, pk_{\bar{\beta}})$.
- $\Xi^*_{1,i''}(m_{\beta,i''}, pk_{\bar{\beta}}, s_{\bar{\beta},i''-1})$ outputs $c_{\bar{\beta},i''} \leftarrow E.\text{Enc}(pk_{\bar{\beta}}, m_{\bar{\beta},i''})$.
- Both parties update their state to contain the transcript of all protocol messages exchanged so far.

After the $r'$ exchange-rounds, $P_{\bar{\beta}}$ computes $c^{**}$ as follows:

\[
c^{**} = \text{Ext}(S, c_{\beta,r+1}) || \ldots || \text{Ext}(S, c_{\bar{\beta},r+r'})
\]

Then, $P_{\bar{\beta}}$ outputs $\mu' \leftarrow \text{SKE.SDec}(sk^*, c^{**})$. (That is, $\Xi^*_{1,\text{out}}(s_{\beta,r'}) = \mu'$.)

5.4.4 Proof that $(\Phi^*, \Xi^*)$ is a subliminal communication scheme

Finally, we give the proof of our main theorem. We recall the statement here.

**Theorem 5.4.1.** Assume there exists a pseudorandom key-exchange protocol. Then there is a multi-message subliminal communication scheme $(\Phi^*, \Xi^*)$, given in Definition 5.4.15.

**Proof.** We define three hybrids.

- **HYBRID 0** ("REAL WORLD")**: Parties execute $(\Phi^*, \Xi^*)$.
- **HYBRID 1**: Exactly like Hybrid 0, except that the seed $S$ in Phase 1a is replaced by a truly random $d$-bit string (the same string for both parties).
• **HYBRID 2:** Exactly like Hybrid 1, except that the key exchange messages $\psi_{b,j}$ in $\Phi^*$ are replaced by random strings. That is, Line 12 is replaced by:

$$P_b \text{ samples } \psi_{b,j} \leftarrow \{0,1\}^\ell \text{ at random.}$$

• **HYBRID 3:** Exactly like Hybrid 2, except that the ciphertexts of SKE in $\Xi^*$ are replaced by random strings. That is, Line 27 is replaced by:

Sample $c^* \leftarrow \{0,1\}^\ell$ at random.

Hybrids 0 and 1 are indistinguishable by direct application of Theorem 5.4.8.

Hybrids 1 and 2 are indistinguishable by the pseudorandomness of the key-exchange protocol (defined in Section 5.1.1). Note that it is essential that the $\Psi$-transcript’s indistinguishability from random holds even in the presence of the established key $sk^*$, since in our protocol $(\Phi^*,\Xi^*)$, the later protocol messages are produced as a function of $sk^*$.

Hybrids 2 and 3 are indistinguishable because ciphertexts of SKE are pseudorandom in the absence of the corresponding secret key $sk^*$. Because we already replaced the messages of $\Psi$ with random messages independent of $sk^*$, the distribution of all protocol messages in Hybrid 1 can be generated based on just the SKE-ciphertexts $c^*$ (of line 27).

Finally, Hybrid 3 is indistinguishable from the ideal distribution

$$\text{Ideal}(pk_0,sk_0,pk_1,sk_1,M)$$

from Definition 5.2.4 by Theorem 5.4.12. Note that the rejection sampler $\Sigma^{E\mu_d}$ of Algorithm 10 has a truly random $d$-bit seed, so to invoke Theorem 5.4.12 we rely on the fact that $S$ is truly random in Hybrid 3.

5.4.5 **On the setup cost and asymptotic rate of $(\Phi^*,\Xi^*)$**

**Setup Cost.** The setup cost of our scheme can be broken down into the costs of Step 1a and Step 1b as follows.

- **Step 1a:** If our scheme is instantiated with the extractor Ext from Propos-
tion 5.4.13, then we need to establish a seed of length $O(\log \lambda)$, which implies that $\tilde{O}(\log \lambda)$ exchange-rounds are required in Step 1a. This is arguably the least efficient step in our scheme; this inefficiency stems from the use of the GT extractor which only outputs one bit: to the best of our knowledge this is the only extractor which applies to our setting. In Section 5.5, we discuss ways in which this cost can be reduced under additional assumptions on the next-message distribution or the encryption scheme $E$ by replacing GT by extractors with longer outputs.

- **Step 1b**: The cost of this step is $k \cdot \frac{\xi}{v}$, where $k$ is the number of rounds of the key-exchange protocol $\Psi$ that we use, $\ell$ is the length of messages in $\Psi$ and $v$ is the output length of $\text{Ext}$. (Concretely, the Diffie-Hellman key exchange protocol achieves $k = 1$ and $\ell = O(\lambda)$.) If we use the extractor from Proposition 5.4.13 as $\text{Ext}$, then we can achieve $v = c \log \lambda$ for any $c > 0$. This implies that the cost of Step 1b is upper-bounded by $c' \frac{\lambda}{\log \lambda}$ for any $c' > 0$. Note that because the min-entropy of ciphertexts from $E$ can be as small as $\omega(\log \lambda)$ and is *a priori* unknown to the designer of the subliminal scheme, the output of $\text{Ext}$ must be $O(\log \lambda)$.

Asymptotic Rate. The asymptotic rate of our scheme depends on the output length of $\text{Ext}$. Using the extractor from Proposition 5.4.13, we can subliminally embed $c \log \lambda$ random bits per ciphertext of $E$ and hence achieve an asymptotic rate of $c \log \lambda$ for any $c > 0$. As in the discussion regarding the cost of Step 1b, this is optimal given that the min-entropy of ciphertexts can be as small as $\omega(\log \lambda)$.

Trade-off Between Running Time and Rate. Note that the parameter $c$ from the previous paragraph controls the trade-off between the running time of our scheme and its asymptotic rate. Indeed, the expected running time of the rejection sampler defined in Algorithm 10 is $O(\lambda^c)$, when embedding $c \log \lambda$ random bits. This trade-off is inherent to rejection sampling, and it is an interesting open question to determine

\[155\] Suppose not, i.e., suppose that the output of $\text{Ext}$ were $v \in \omega(\log \lambda)$ bits long. Then the adversary could choose an encryption scheme whose ciphertexts have min-entropy $z \in \omega(\log \lambda) \cap o(v)$ (e.g., $z = \sqrt{v(\lambda) \log \lambda}$). Since the extractor output cannot have more min-entropy than its input, the extractor's output when evaluated on ciphertexts would be distinguishable from random $v$-bit strings.
whether it can be improved by an alternative technique.

5.5 Improving setup cost

In this section, we present alternative constructions which improve the setup cost of subliminal communication under additional assumptions, either on the next-message distribution $\mathcal{M}$, or on the public-key encryption scheme $E$. These additional assumptions allow us to replace the “greater-than” extractor in Step 1a of Definition 5.4.2 with extractors with longer output, thus reducing the number of exchange-rounds required to establish the shared seed. Section 5.5.1 gives a construction when $E$ is “succinct” (i.e., has a constant expansion factor). Section 5.5.2 presents a construction when $\mathcal{M}$ has a known amount of min-entropy. Both these constructions yield a seed establishment in two exchange-rounds.

5.5.1 Succinct encryption schemes

Let us suppose that the adversarially chose scheme $E$ is succint. Here we define succint as having an expansion factor less than 2. Recall that the expansion factor is defined to be the ratio of ciphertext length to plaintext length. Under this assumption, we can improve the number of rounds in the construction of Section 5.4.1 by replacing the extractor GT by the extractor BLE from [64]. Note that this extractor requires the min-entropy rate of the sources to be slightly above $\frac{1}{2}$, yet ciphertexts of $E$ only have min-entropy $\omega(\log \lambda)$. However, the succinctness assumption combined with semantic security implies that ciphertexts have sufficiently large HILL entropy: they are computationally indistinguishable from sources of min-entropy rate slightly above $\frac{1}{2}$. Since the extractor is a polynomial time algorithm, its output when computed on ciphertexts from $E$ will be computationally indistinguishable from the uniform distribution. Formally, we prove the following theorem.

**Theorem 5.5.1.** Let us denote by $n$ (resp. $p$) the bit-lengths of ciphertexts (resp. plaintexts) from $E$. Let BLE be the function constructed in [64] and let us denote by
\[ B : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^v \] its truncation to \( v = O(\log \lambda) \) bits. With the same notations as Theorem 5.4.8, define:

\[ S = (B(C_{0,1}, C_{0,2}), B(C_{1,1}, C_{1,2}), \ldots, B(C_{0,2^p-1}, C_{0,2^p}), B(C_{1,2^p-1}, C_{1,2^p})). \]

There is a negligible function \( \varepsilon \) such that \( \| (K_0, K_1, S) - (K_0, K_1, U_{2^p}) \|_s \leq \varepsilon. \)

**Proof.** For any \( \gamma > 0 \), the function \( \text{BLE} \) from [64] can extract \( 2p - n - \gamma p \) random bits with bias \( \varepsilon = 2^{-1/2^\gamma} \) from two independent \( n \)-bit sources of min-entropy \( p \). Note that for \( n \leq (1 - \gamma)2p, 2p - n - \gamma p \geq \gamma p \), and by semantic security of \( E \), we have that \( n \) (and hence \( p \)) are super-logarithmic. Hence the function \( B \) defined in Theorem 5.5.1 is well-defined and has bias \( \varepsilon \in \text{negl} \) for independent sources of min-entropy \( p \).

For public key \( pk \), define \( Y^{pk} = E.\text{Enc}(pk, U_p) \) the random variable obtained by encrypting a message chosen uniformly at random. By correctness of \( E \), \( Y^{pk} \) is a flat source supported on \( \{0,1\}^p \), hence \( H_\infty(Y^{pk}) = p \). Hence for \( Z^{pk} \) an independent copy of \( Y^{pk} \), we have \( \| B(Y^{pk}, Z^{pk}) - U_v \|_s \leq \varepsilon(\lambda) \). By semantic indistinguishability of \( E \), we have that for any ciphertext \( C \) of \( E \), \( C \approx Y^{pk} \); i.e., \( C \) has HILL-entropy \( p \).

Adapting the proof of Lemma 5.4.6 and using the same notation, since \( v = O(\log \lambda) \), we get \( \| (PK, SK, B(C_0^{PK}, C_1^{PK})) - (PK, SK, U_v) \|_s \leq \varepsilon \). From there, the proof of Theorem 5.5.1 follows verbatim the proof of Theorem 5.4.8. \( \square \)

**Remark 5.5.2.** Theorem 5.5.1 implies in particular that a \( \tilde{O}(\log \lambda) \) random seed can be established in step 1a of our subliminal scheme in only two exchange-rounds of communication, whereas the construction of Theorem 5.4.8 with the GT extractor requires \( \tilde{O}(\log \lambda) \) exchange-rounds.

### 5.5.2 Next-message distributions with min-entropy

The previous subsection reduced the number of rounds to establish the shared seed by using a two-source extractor with more than one bit of output (unlike the GT extractor). However, this required \( E \) to be succinct to guarantee that ciphertexts of \( E \) have sufficient HILL-entropy. In this section, we observe that if the message distribution
itself has enough min-entropy, then we can use the plaintext and corresponding ciphertext as a pair of sources to extract from, and obtain a similar improvement without requiring that $E$ be succinct. Note that this construction again exploits the semantic security of $E$ to guarantee that the plaintext and the ciphertext are indistinguishable from independent sources to the two-source extractor.

All we require of $E$ in this subsection is that the ciphertext distribution has min-entropy at least $\omega(\log \lambda)$, which follows from semantic security (Lemma 5.4.4); however, we additionally require that the next message distribution $M$ have min-entropy rate above $\frac{1}{2}$. While this is quite a lot of min-entropy to demand from $M$, we note that the precise requirement of $\frac{1}{2}$ arises from the current state of the art in two-source extractors, and improved two-source extractor constructions would directly imply improvements in the min-entropy requirement of our construction. Recent research in two-source extraction has been quite productive; with luck, future advances will provide us with an alternative which demands much less entropy from $M$.

**Theorem 5.5.3.** Suppose that $k_1$ is such that for all $k_2 = \omega(\log \lambda)$ there exists a negligible function $\varepsilon$ and a $(k_1, k_2, \varepsilon)$-two source extractor $2\text{Ext}$ with output length $\ell = O(\log \lambda)$, and let $\rho \in \text{poly}$. Define random variables as follows.

- For $b \in \{0, 1\}$, let $K_b = (PK_b, SK_b) = E.\text{Gen}(1^\lambda)$.
- For $b \in \{0, 1\}$ and $i \in [\rho]$, let $M_{b,i}$ and $C_{b,i} = E.\text{Enc}(PK_{1-b}, m_{b,i})$ be the messages and ciphertexts exchanged between $P_0$ and $P_1$ in $\rho$ exchange-rounds.
- $S = (2\text{Ext}(M_{0,1}, C_{0,1}), 2\text{Ext}(M_{1,1}, C_{1,1}), \ldots, 2\text{Ext}(M_{0,\rho}, C_{0,\rho}), 2\text{Ext}(M_{1,\rho}, C_{1,\rho}))$.

Then if the next message distribution $M(\text{conv})$ has min-entropy at least $k_1$:

$$\|(K_0, K_1, S) - (K_0, K_1, U_\rho)\|_\infty \leq \varepsilon.$$

**Proof.** Let us consider $M$ and $M'$ two independent samples from the next-message distribution and let us denote by $C_{M}^{PK}$ and $C_{M'}^{PK}$ their encryption under key $PK$. Consider a two-source extractor $2\text{Ext}$ as in the statement of the theorem. By semantic
security, it follows that for some negligible function $\varepsilon'$:

$$2\text{Ext}(M, C^{PK}_M) \approx \varepsilon' \cdot 2\text{Ext}(M', C^{PK}_M).$$

By property of $2\text{Ext}$ it follows that $\|2\text{Ext}(M', C^{PK}_M) - U_\ell\|_s \leq \varepsilon$. Since $\ell = O(\log \lambda)$, we can prove similarly to Lemma 5.4.6:

$$\|(PK, SK, 2\text{Ext}(M, C^{PK}_M)) - (PK, SK, U_\ell)\|_s \leq \varepsilon + \varepsilon'.$$

We can now conclude similarly to the proof of Theorem 5.4.8. \hfill \Box

The following result from [153] implies that for any $\delta > 0$, when $k_1 = \frac{1}{2} + \delta$, two-source extractors exist which may be used in the above theorem. The output length of such extractors is logarithmic, implying that a $\tilde{O}(\log \lambda)$-bit random seed $S$ can be established in two exchange-rounds of communication.

**Lemma 5.5.4** (Theorem 1 in [153]). For any $\delta > 0$, $k_1 \geq \frac{1}{2} + \delta$ and $k_2 = \omega(\log \lambda)$, there exists a negligible function $\varepsilon$ and a $(k_1, k_2, \varepsilon)$-two source extractor with output length $\ell = O(\log \lambda)$.

### 5.6 Steganographic communication schemes

In this section, we give a formal definition of a steganographic communication scheme, formalized as a protocol consisting of $r$ exchange-rounds in which two parties $P_0$ and $P_1$ engage in communication distributed according to a cover distribution $C$, and in which $P_0$ steganographically transmits a hidden message $\mu$ to $P_1$. The cover distribution $C$ is a parameter of the communication scheme.

Based on this definition, we finally formalize and prove the impossibility result of Theorem 5.3.3 from Section 5.3.2.

**Definition 5.6.1** (Steganographic communication scheme). A steganographic com-
munication scheme with respect to a cover distribution $\mathcal{C}$ is a two-party protocol:

$$\Pi^C = (\Pi^C_{0,1}, \Pi^C_{1,1}, \Pi^C_{0,2}, \Pi^C_{1,2}, \ldots, \Pi^C_{0,r}, \Pi^C_{1,r}, \Pi^C_{1,\text{out}})$$

where $r \in \text{poly}$ and where each $\Pi^C_{b,i}$ is a PPT algorithm with oracle access to a $\mathcal{C}$-sampler. Party $P_0$ is assumed to receive as input a message $\mu$ that is to be conveyed to $P_1$ in an undetectable fashion. The protocol must satisfy the following syntax, correctness and security guarantees.

- **Syntax.** For each $i = 1, \ldots, r$:
  1. $P_0$ draws $(c_{0,i}, s_{0,i}) \leftarrow \Pi^C_{0,i}((\mu, s_{0,i-1}))$, locally stores $s_{0,i}$, and sends $c_{0,i}$ to $P_1$.
  2. $P_1$ draws $(c_{1,i}, s_{1,i}) \leftarrow \Pi^C_{1,i}(s_{1,i-1})$, locally stores $s_{1,i}$, and sends $c_{1,i}$ to $P_0$.

After the $r$th exchange-round, $P_1$ computes $\mu' = \Pi^C_{1,\text{out}}(c_{0,1}, c_{1,1}, \ldots, c_{0,r}, c_{1,r})$ and halts.

- **Correctness.** For any $\mu \in \{0,1\}^\lambda$, $\mu' = \mu$ with overwhelming probability. The probability is taken over the randomness of all of the $\Pi^C_{b,i}$ and their $\mathcal{C}$-samples.

- **Steganographic Indistinguishability w.r.t. $\mathcal{M}$.** The following distribution

$$\{(c_{0,1}, c_{1,1}, \ldots, c_{0,r}, c_{1,r}) : \mu \leftarrow \mathcal{M}, (c_{0,i}, s_{0,i}) \leftarrow \Pi^C_{0,i}((\mu, s_{0,i-1})), (c_{1,i}, s_{1,i}) \leftarrow \Pi^C_{1,i}(s_{1,i-1})\}$$

is computationally indistinguishable from the cover distribution of length $2r - 1$.

(Note that $\mathcal{C}_{2r-1}$ is not necessarily a product distribution.)

Depending on the specific application at hand, a steganographic communication scheme might require that steganographic indistinguishability hold w.r.t. all message distributions $\mathcal{M}$, or alternatively, only require that it hold w.r.t. certain specific message distributions (e.g., uniformly random messages).

Definition 5.6.1 concerns the transmission of only a single message. There is a natural generalization of this definition to a more general multi-message version, analogous to how Definition 5.2.4 is a multi-message generalization of Definition 5.2.2 in the context of subliminal communication schemes. Much as observed in Remark 5.2.7
in the context of subliminal communication schemes, the natural multi-message generalization of Definition 5.6.1 is equivalent to Definition 5.6.1 in the sense that the existence of any single-message scheme easily implies a multi-message scheme and vice versa; the multi-message version of the definition is mainly useful when considering the asymptotic efficiency of schemes. We present only Definition 5.6.1 here as the simpler single-message definition suffices in the context of an impossibility result.

The existing notions of public-key steganography and steganographic key exchange, as well as the subliminal communication schemes introduced in this work, are all instantiations of the more general definition of a (multi-message) steganographic communication scheme.\textsuperscript{16}

5.6.1 Proof of Theorem 5.3.3

In this subsection we give the proof of impossibility of non-trivial steganographic communication in the presence of an adversarially chosen cover distribution. We recall the statement of Theorem 5.3.3 from Section 5.3.2.

\textbf{Theorem 5.3.3.} Let $\Pi$ be a protocol with the syntax of a steganographic communication scheme. Then for any $k \in \mathbb{N}$, there exists a cover distribution $C$ of conditional min-entropy $k$ such that steganographic indistinguishability of $\Pi$ does not hold.

\textit{Proof.} The proof exploits the impossibility of extracting more than one bit from Santha-Vazirani (SV) sources (Proposition 6.6 in [175]). Recall that a $\delta$-SV source $X \in \{0,1\}^n$ is defined by the following conditions:

$$\delta \leq \Pr[X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq 1 - \delta$$

for all $i \in [n]$ and all $(x_1, \ldots, x_{i-1}) \in \{0,1\}^{i-1}$.

\textsuperscript{16}Public-key steganography protocols can be thought of as being parametrized by the public/private key pairs of the communicating parties, which are initially sampled by the parties according to some key generation algorithm. However, the definition of a steganographic communication protocol as stated is not parametrized in an analogous way. This syntactic discrepancy can be resolved by equivalently thinking of the public-key steganography as a family of steganographic communication protocols each of which has the key pairs hardwired, induced by the sampling of key pairs according to the key generation algorithm; and then requiring the steganographic indistinguishability guarantee to hold only with overwhelming probability over key generation.

161
Consider $\delta > 0$ and $\Pi$ a steganographic communication scheme with $r$ exchange-rounds. Let $n \in \mathbb{N}$ be such that $\delta$-SV sources of length $n$ have conditional min-entropy $k$ when interpreted as $2r - 1$ blocks of size $\frac{n}{2r-1}$. Assume for contradiction that $\Pi$ can embed at least two messages (say, 0 and 1). Steganographic indistinguishability applied to the case where $\mu \leftarrow \{0, 1\}$ implies the existence of a negligible $\varepsilon$ such that $1[\Pi_{1,\text{out}}(C_{2r-1}) = 0]$ is a random bit with bias at most $\varepsilon$.

But Proposition 6.6 in [175] implies the existence of a $\delta$-SV source $C_{2r-1}$ such that $\Pr[\Pi_{1,\text{out}}(C_{2r-1}) = 0] \geq 1 - \delta$ or $\Pr[\Pi_{1,\text{out}}(C_{2r-1}) = 0] \leq \delta$. In other words, $1[\Pi_{1,\text{out}}(C_{2r-1}) = 0]$ has bias at least $\frac{1}{2} - \delta$. For $\delta < \frac{1}{2} - \varepsilon$, this is a contradiction. □

**Remark 5.6.2.** At first glance, it may seem that one could bypass this impossibility and communicate more than one bit of information by sequentially repeating the same steganographic scheme. However, one could then apply Theorem 5.3.3 to the sequential composition and obtain a new cover distribution which would break this scheme.

### 5.7 Open problems

We conclude the chapter with a brief discussion of open problems that would be interesting directions for future work.

**Deterministic Extraction.** Our impossibility result in Theorem 5.3.1 holds because the adversary can choose the encryption scheme $E$ as a function of a given candidate subliminal scheme. However, note that under the additional assumption that $E$ is restricted to a predefined class $E'$ of encryption schemes, we could bypass this impossibility as long as a deterministic extractor that can extract randomness from ciphertexts of any encryption scheme in $E'$ exists. We are only aware of two deterministic extractors leading to a positive result for restricted classes of encryption schemes:

- if an upper bound on the circuit size of $E$ is known, then we can use the deterministic extractor from [174]. This extractor relies on strong complexity-theoretic...
assumptions and requires the sources to have min-entropy \((1 - \gamma)n\) for some unspecified constant \(\gamma\).

- If \(E\) is computed by a circuit of constant depth \((AC^0)\), then the deterministic extractor of [176] can be used and requires \(\sqrt{n}\) min-entropy.

Note that both these extractors have a min-entropy requirement which is too strict to be directly applicable to ciphertexts of arbitrary encryption schemes, but it would be interesting to give improved constructions for the specific case of encryption schemes. This would also have direct implications for the efficiency of the subliminal scheme we construct in Section 5.4.2: indeed, one could then skip Step 1a and use a deterministic extractor directly in Steps 1b and 2, thus saving \(\tilde{O}(\log \lambda)\) exchange-rounds in the setup phase.

**Multi-Source Extraction.** Another interesting question is whether multi-source extractors for the specific case when the sources are independent and identically distributed can achieve better parameters than extractors for general independent sources. We already saw that a very simple extractor (namely, the “greater-than” function) works for i.i.d. sources and extracts one bit with negligible bias, even when the sources only have \(\omega(\log \lambda)\) min-entropy. The non-constructive result of [53] guarantees the existence of a two-source extractor of negligible bias and output length \(\omega(\log \lambda)\) for sources of min-entropy \(\omega(\log \lambda)\). However, known explicit constructions are far from achieving the same parameters, and improving them in the specific case of identically distributed sources is an interesting open problem which was also mentioned in [17].

**Acknowledgements for work presented in this chapter.** We are grateful to Omer Paneth and Adam Sealfon for insightful remarks on an earlier draft of this work.
Chapter 6

Repudiability and (Un)claimability of Ring Signatures

This chapter presents results from a paper titled "It Wasn’t Me! Repudiability and (Un)claimability of Ring Signatures" [149] that I coauthored with Adam Sealfon.

6.1 Introduction

Ring signatures, introduced by [154], are a variant of digital signatures which certify that one among a particular set of parties has signed a particular message, without revealing which specific party is the signer. This set is referred to as a “ring.” Ring signatures can be useful, for example, to certify that some leaked information comes from a privileged set of government or company officials without revealing who the whistleblower is, or to issue important orders or directives without setting up the signing individual to be made a scapegoat for repercussions.

In a ring signature scheme, just as in a traditional digital signature scheme, any party can create a key pair for signing and verification, and publish it without in-

---

1The paper is in submission at the time of writing.
2When it comes to national security issues, for instance, there is a well-known reluctance of law-makers to “roll back” existing laws or reduce checking or surveillance measures — even if such measures are deemed likely to be beneficial — due to the risk of ending up a scapegoat (and losing one’s job) upon any future terrorist attack.
teracting with any other parties. Signers can produce signatures that verify with respect to any set of verification keys (which must include their own), and unforgeability guarantees that no party can produce a valid signature with respect to a set of verification keys, for none of which he knows a corresponding secret key.

Now let us consider a hypothetical tale, wherein two candidates Alice and Bob are running for president in the land of Oz. Oz is notorious for its petty partisan politics and its politically disengaged citizenry wont to vote for whomever appears friendlier in a series of nationally televised teeth-baring contests between the main-party candidates. At the peak of election season, a disgruntled citizen Eve decides to help out her preferred candidate Bob by publishing the following message, which goes viral on the social networks of Bob supporters:

*I created a notorious terrorist group and laundered lots of money!*

_Signed: Alice or Eve or Alice’s campaign chairman._

Of course, the virally publicized message does not actually incriminate Alice at all, since any one of the signatories could have produced it. However, perhaps there is nothing that Alice can do to stay the doubt in the minds of her suspicious skeptics. Ring signatures are designed to allow anyone to attach anyone else’s name to a signature. But what guarantee do they provide if a purported signatory wishes to denounce a signed message? Or if the real signatory wishes to later reveal that it was indeed he or she who signed a particular message (perhaps a possibility in whistleblowing scenarios, for example)?

The answer is: it depends. The most detailed taxonomy of security definitions for ring signatures was given by [33], which presents a series of anonymity guarantees of increasing strength. A natural anonymity guarantee defined by [33], called “anonymity against adversarially chosen keys,” is informally described as follows: an adversary who controls all but \( k \geq 2 \) parties in a ring, and who may produce his own malformed key pairs as well as corrupt honest parties’ keys, must have negligible advantage at guessing which of the \( k \) honest parties produced a given signature. Observe that this anonymity definition could allow for a party to ascertain whether or not a given
signature was produced by her own signing key, and perhaps also to convince others of this fact — but does not guarantee either of these capabilities. A guarantee one way or the other seems highly desirable.

On the other hand, the strongest of the anonymity definitions of [33] (called “anonymity against full key exposure”) requires that even if an adversary compromises every single party in a ring, the adversary cannot identify the signers of past signatures. It is relatively straightforward to see that under such a strong anonymity guarantee, Alice would have no way to convince anyone that she did not produce the objectionable message; indeed, she cannot tell the difference between a signature produced using her own signing key and one produced using someone else’s. Intriguingly, while denouncing a signature is guaranteed to be impossible under anonymity against full key exposure, it could still be that the true signer could credibly reveal their identity and claim the signature: for example, by remembering the signing randomness.

The ability to identify whether one’s own signing key was used to produce a particular signature can be a feature or a bug. To protect anonymity of past signatures against a very strong adversary who might compromise all the secret keys in a ring, it seems beneficial to disallow distinguishing between one’s own signatures and someone else’s. On the other hand, without the ability to distinguish between these, it would be virtually impossible to tell if someone had stolen your signing key, in a ring signature scheme. Moreover, in certain circumstances, it could be beneficial for members of a ring to have the ability to disown signatures of messages that they wish to denounce. Conversely, the signer of a message might wish, in certain circumstances, to later prove to the world that he was the one who produced a particular signature in the past.

In the preceding discussion, we have identified the four possibly useful notions of repudiability, unrepudiability, claimability, and unclaimability of ring signatures. Our contributions in this paper are both in the form of new definitions and constructions of these notions, as summarized next.
Our contributions in summary. We formalize the notions of repudiability, claimability, and unclaimability of ring signatures, as well as strengthened anonymity definitions which are compatible with these notions. We observe that unclaimability implies unrepudiability (in a nutshell, because a failed repudiation can be used as a claim). Definitionally, we show that repudiability and claimability are compatible with anonymity against adversarially chosen keys, while the opposite goal of unclaimability is implied by anonymity against full key exposure.

We provide three constructions based on different assumptions, one for each of the three notions: repudiability, claimability, and unclaimability. Perhaps the most surprising of these is unclaimability: a natural first intuition is that meaningful notions of unclaimability might be impossible to achieve, since a signer can always remember the signing randomness (and later present it as “proof” of having produced a signature). The key insight for our definition and construction of unclaimable ring signatures is that the signing randomness does not constitute a convincing claim if anyone can produce credible signing randomness for any signature in which they are implicated: i.e., if everyone in the ring can produce credible signing randomness. Our construction of unclaimable ring signatures is an augmentation of the lattice-based ring signature scheme of [44] by adding additional algorithms; the property that anyone can produce credible signing randomness is achieved via lattice trapdoors.

Our construction of repudiability ring signatures is based on trapdoor permutations (i.e., the same assumptions as required by the [33] construction). It does not use standard ring signatures as a building block, and as such can also be viewed as a new construction of ring signatures (with, however, the same assumptions and similar efficiency to [33]). Our construction of claimable ring signatures, on the other hand, is a generic black-box transformation from any standard ring signature scheme to a claimable one. We overview our contributions in more detail below.

6.1.1 Definitional contributions

Repudiability. We define a repudiability ring signature scheme as a ring signature scheme which is equipped with additional algorithms Repudiate and VerRepud, that
work as follows. **Repudiate** takes as input a signing key \( sk \), a ring signature \( \sigma \), and a "ring" \( R \) (i.e., a set of verification keys), and outputs a repudiation \( \xi \). **VerRepud** takes as input a ring \( R \), a signature \( \sigma \), a repudiation \( \xi \), and a verification key \( vk \), and outputs a a single bit indicating whether or not \( \xi \) is a valid repudiation attesting that \( \sigma \) was not produced by \( vk \). The two requirements for a ring signature scheme to be repudiable are, informally, as follows.

1. **Correctness:** Any member of a ring must be able to produce valid repudiations of any signature that he did not produce.
2. **Soundness:** A cheating signer must not be able to produce a valid signature with respect to a ring, and also be able to produce valid repudiations of that signature under every verification key in that ring that he owns.

Once a ring signature scheme is equipped with these additional repudiation algorithms, the standard definitions of unforgeability and anonymity against adversarially chosen keys are insufficient to capture the natural guarantees that would be desired for a repudiable ring signature scheme. In particular, we would not like repudiations to compromise the unforgeability or anonymity of any future signatures. Accordingly, we modify the definitions of unforgeability and anonymity for repudiable ring signatures (Definitions 6.3.4 and 6.3.5), by additionally giving the adversary access to a repudiation oracle. This ensures that repudiations of past signatures do not affect the security guarantees of future signatures. See Section 6.3.1 for formal definitions of repudiability.

**Claimability.** We define a claimable ring signature scheme as a ring signature scheme which is equipped with additional algorithms **Claim** and **VerClaim**, that work as follows. **Claim** takes as input a signing key \( sk \), a signature \( \sigma \), and a ring \( R \), and outputs a claim \( \zeta \). **VerClaim** takes a input a ring \( R \), a verification key \( vk \), a signature \( \sigma \), and a claim \( \zeta \), and outputs a single bit indicating whether or not \( \zeta \) is a valid claim attesting that \( \sigma \) was produced by \( vk \). Similarly to above, we have two natural requirements for a claimable ring signature scheme, stated informally next.
1. **Correctness:** Any honest signer must be able to produce a valid claim with respect to any signature that he produced.

2. **Soundness:** No adversary can produce a valid claim with respect to a signature produced by an honest signer, even if the adversary can choose the message and ring with respect to which the signature is produced, and can insert malformed verification keys into the ring.

As above, once a ring signature scheme is equipped with these additional claiming algorithms, the standard definitions of *unforgeability* and *anonymity* against adversarially chosen keys are insufficient. We modify the definitions of anonymity and unforgeability for claimable ring signatures (Definitions 6.3.9 and 6.3.10), by additionally giving the adversary access to a *claim oracle*. See Section 6.3.3 for formal definitions of repudiability.

*Repudiability* and *claimability* are compatible, i.e., a ring signature scheme could have both properties simultaneously. Indeed, our repudiable and claimable constructions together give rise to such a scheme. However, it is important to note that the unforgeability and anonymity definitions corresponding to a repudiable-and-claimable ring signature scheme are *not* the conjunction of unforgeability and anonymity for repudiable ring signatures and for claimable ring signatures. Rather, the unforgeability and anonymity definitions for a repudiable-and-claimable ring signature scheme involve an adversary which is given access to both a repudiation oracle and a claim oracle at once. See Section 6.3.5 for further discussion on repudiable-and-claimable schemes.

**Unclaimability.** We also introduce the notion of an *unclaimable* ring signature scheme, in which the signer *provably cannot* convincingly claim that she was the one who produced the signature. At first glance, this may seem like an impossible goal. The signer can save the signing randomness, and reveal it along with her secret key as a “proof” that she was the one to generate the signature. While the signer can always do this, it is not necessarily true that this constitutes a convincing proof of authorship of the signature. In particular, it may be possible for other members of
the ring to take a valid signature and produce fake randomness that produces the desired signature using their own signing keys.\footnote{This is intuitively somewhat similar to the goal in deniable encryption \cite{deniable}, in which the encryptor can produce fake randomness to explain a particular ciphertext as an encryption of an arbitrary message.}

Consequently, we define an \textit{unclaimable} ring signature scheme to be one in which any member of the ring can produce fake signing randomness for a signature that is distributed indistinguishably from real signing randomness. We consider a strong flavor of this definition in which the indistinguishability property is statistical. Informally, we require the following property of an unclaimable ring signature scheme.

1. \textit{Indistinguishability:} Any member of a ring must be able to produce fake signing randomness given a signature. The signature and fake signing randomness must be distributed statistically close to an honestly generated signature and randomness produced by that individual on the same message, even given all verification keys and signing keys.

We formally define unclaimability in Section 6.3.4. Note that unclaimability also implies anonymity against full key exposure, the strongest form of anonymity defined in \cite{full_key}.

\textbf{Remark 6.1.1.} \textit{Even under this definition, if the signer chooses a message to sign that corresponds to a secret known only to herself, then she may still be able to convince others that she was the signer. For instance, if the signed message is the output of a one-way function, she may be able to convince others that she was the signer by subsequently revealing the preimage. Even more flagrantly, the signed message could contain a signature using a standard (non-ring) signature scheme, directly identifying the signer. This property is rather inherent: if knowledge of the contents of the message itself at the time of signing are enough to identify the signer, then no security property on the signature scheme can enforce that the signer remains hidden, since the identification of the signer is unrelated to the signature and based only on the signed message.}

\textit{Indeed, ring signatures were not designed to provide anonymity for signers who
want to identify themselves, but rather for those who desire anonymity. Similarly, our unclaimability definition does not guarantee unclaimability for those who want to identify themselves, but rather provides credibility for a signer who wants to later be able to claim (e.g., under duress) that she could not convincingly claim the signature even if she wanted to. In particular, even an adversary with unlimited computational power who obtains the secret keys belonging to every member of the ring and a purported signing randomness from an alleged signer, he still will not be convinced of the identity of the signer, since fake signing randomness from the right distribution can be produced for every member of the ring.

**Unrepudiability.** Unclaimability intuitively guarantees that no member of the ring can convincingly prove that she was the signer. A related, weaker notion might be desirable in some circumstances is that of unrepudiability, which guarantees that no member of the ring can convincingly prove that she was not the signer. Informally, such a guarantee could be characterized as follows: any computation that can be performed upon a (valid) signature and a key pair corresponding to a non-signer, can be simulated by a computation on that signature and a key pair belonging to the signer. Unrepudiability is implied by anonymity against full key exposure and by unclaimability.

### 6.1.2 Overview of our constructions

**A first attempt at repudiability.** Suppose that each key pair in our ring signature scheme includes a public-key encryption key pair. A natural first attempt at constructing a repudiable and claimable ring signature scheme is as follows: a signature with respect to ring \( R = \{v_{k_1}, \ldots, v_{k_N}\} \) contains \( N \) ciphertexts \( c_{t_1}, \ldots, c_{t_N} \) where each \( c_{t_i} \) is an encryption under the public encryption key \( p_{k_i} \) corresponding to \( v_{k_i} \). The ciphertext corresponding to the signer’s identity must encrypt the value 1, while all the other ciphertexts must encrypt the value 0. Then, to repudiate a signature, one can show that the ciphertext corresponding to one’s verification key

---

4This is the case in the construction of [33].
decrypts to the value 0; and to \textit{claim} a signature, one can show that the corresponding ciphertext decrypts to the value 1.\footnote{The value of the decryption could be demonstrated verifiably without revealing the secret decryption key, e.g., by using a scheme in which the encryption randomness is recoverable given the decryption key, and presenting the encryption randomness as proof of the decrypted value.}

What goes wrong? The proposed scheme does not seem to violate anonymity, since there is one ciphertext per identity in the ring, and the semantic security of the encryption scheme ought to hide the encrypted values which indicate the signer’s identity. Moreover, non-signers can only decrypt their own ciphertexts, so can learn only that they are not the signer (i.e., they cannot learn the identity of the signer). In fact, the problem is not with anonymity, but repudiability. As described, the scheme would allow for a malicious signer to encrypt whatever values he wants, rather than the required sequence of 0s and a 1. Moreover, the required format cannot simply be enforced by requiring the signer to provide a zero-knowledge (or witness-indistinguishable) proof that the encrypted values are as required, for then the statement to be proven would have to ascertain the signer’s identity in order to make sure that the encryption of 1 is at the correct position. This sort of approach might work if we had \textit{proofs of knowledge} as a building block, since then the signer could confirm his identity by proving knowledge of the secret key corresponding to his verification key, but unfortunately, there are no constructions of witness-indistinguishable proofs of knowledge in fewer than three rounds.

In fact, loosely speaking, any approach that requires the signer to “distinguish” his own identity from the others in the ring (in some encrypted fashion) at the time of signing suffers from the problem that malicious signers can deviate from the required signing algorithm, and the correct signature format cannot be enforced by zero-knowledge or witness-indistinguishable proofs since that would require ascertaining the true identity of the signer. Thus, we require a different approach.

\textbf{Our repudiable construction.} Our construction relies on ZAPs (two-round public-coin witness-indistinguishable proofs) and verifiable random functions (VRFs) as building blocks. Our building blocks have some overlap with those of the ring sig-
nature construction of [33], which uses ZAPs, public-key encryption, and a digital signature scheme. Both our scheme and theirs use ZAPs to achieve anonymity of the ring signatures, but with different approaches: the statements proven by the ZAPs are quite unrelated in the two constructions. Moreover, in our scheme, we do not need public-key encryption or a digital signature scheme, and instead use VRFs directly to achieve unforgeability and repudiability — the structure of our construction is thus quite different from that of [33].

At a very high level, each signing key in our construction contains a tuple of four VRF keys. A signature consists of the output of each of the signer’s VRFs on the message, along with a ZAP proof that (several of) the VRF values in the signature are correct w.r.t. the VRF verification key of some member of the ring. A repudiation for individual $i$ consists of a ZAP proof that some of the VRF values in the signature are different from the correct values for party $i$’s VRFs evaluated at the message. One complication arises because we must guarantee that the release of a repudiation for individual $i$ on a message does not subsequently allow a different member of the ring to produce a signature on the message that cannot be repudiated by individual $i$. We get around this difficulty by relying on the witness indistinguishability property of the ZAP and ensuring that the repudiation does not reveal the actual VRF outputs of the repudiator; that is, the ZAP proof is produced with the VRF proof as a witness. The specific statement proven by the ZAPs is that some specific combination of at least two of the purported VRF outputs is correct. Although in the honest usage of the scheme, all four are produced correctly, we design the specific structure of the statements proved in order to allow a hybrid argument to argue indistinguishability between signatures of different signers in a ring. This scheme of proving the correctness of VRF outputs turns out also to imply unforgeability, not only repudiability, so we do not need to rely on any underlying signature scheme as building block. (In other words, our scheme can also be seen as a new construction of plain signatures based on VRFs.)
Our claimable construction. Our construction is a generic transformation from any standard (non-ring) signature scheme RS to a claimable one. The construction uses commitment schemes, standard signatures, and PRFs (which are all achievable from one-way functions). The basic idea is to take a signature $\sigma_{RS}$ under RS and append to it a commitment $c$ to $(vk, \sigma_{RS})$ where $vk$ is the verification key of the signer. The verification algorithm simply checks whether $\sigma_{RS}$ verifies. The claim consists of a decommitment revealing that $c$ is a commitment to $(vk, \sigma_{RS})$. Intuitively, by the hiding property of the commitment scheme, the identity of the signer is hidden until he chooses to publish a claim.

The simple transformation just described runs into a couple of problems when examined in detail. First, what if a signer commits to $(0_{RS}, vk')$ where $vk'$ is not his own key but that of someone else in the ring? This ability would violate equation (6.7) of Definition 6.3.8 (claimability). To prevent such behavior, our construction actually commits to a standard (non-ring) signature on $(vk, \sigma_{RS})$. The unforgeability property of standard signatures then guarantees, intuitively, that a signer cannot convincingly make a claim with respect to any verification key unless he knows a corresponding signing key.

A second hurdle encountered by the scheme thus far described is that the signer must remember the commitment randomness in order to produce a claim. It is preferable that the signer need not be stateful in between signing and claiming; and indeed, recall that Definition 6.3.8 formalizes this property. To resolve this, our construction derives commitment randomness from a PRF. For similar reasons, the signing randomness for the standard (non-ring) signature in our construction is also derived from a PRF.

Our unclaimable construction. Our construction of unclaimable ring signatures is an extension of the SIS-based ring signature scheme of Brakerski and Kalai [44]. The construction is based on trapdoor sampling. In this overview, we describe a simplified version of the scheme. The full scheme is described in Section 6.6. The basic idea for obtaining unclaimability is that each identity corresponds to a public
matrix $A_i \in \mathbb{Z}_q^{n \times m}$ sampled together with a secret trapdoor $T_i$. A signature will consist of short vectors $x_i \in \mathbb{Z}_q^m$ such that

$$\sum A_i x_i = y,$$

where $y$ is a target value. For this overview, we can think of $y$ as the output of a random oracle on the message; in the actual construction, $y$ will be obtained as the sum of additional matrix-vector products. In order to sign the message, signer $i$ first samples short vectors $x_j$ for each $j \neq i$. Then, using the lattice trapdoor $T_i$, he samples a short vector $x_i$ such that the equation

$$x_i = y - \sum_{j \neq i} A_j x_j$$

is satisfied. The signature is the list of vectors $\sigma = (x_i)_i$. Using properties of lattice trapdoors, it follows that the distribution over $(x_i)_i$ can be made to be statistically close no matter which trapdoor was used to produce the signature. Moreover, given a vector $x^*$ to be produced, we can sample random coins that will yield that vector under either the ordinary sampling algorithm or the trapdoor sampling algorithm. Consequently, we obtain an algorithm that can produce explanatory randomness for a signature under any identity in the ring.

Removing the random oracle to obtain ring signatures in the plain model (and unclaimable ones) requires several complications. [44] first describe a basic ring signature scheme with weaker unforgeability properties, in which the target vector $y$ is determined using additional matrix-vector products for matrices that depend on the bits of the message. They then amplify the security of the scheme through a sequence of transformations that ultimately yield a scheme with full unforgeability. In Section 6.6, we first define an algorithm for producing explanatory randomness for their basic scheme, and then describe how to modify this algorithm for each modification of the basic scheme, ultimately yielding an unclaimable ring signature scheme based on the SIS assumption.
6.1.3 Other related work

Since the original proposal of ring signatures by [154], a number of scattered definitional variants and additional guarantees have been proposed. Threshold ring signatures require at least a certain threshold $k$ of ring members to participate, in order to produce a valid signatures. (This follows the more general paradigm of threshold cryptography, introduced by [62].) Linkable ring signatures [128] allow the identification of signatures that were produced by the same signer, without compromising the anonymity of the signer within the ring. An enhancement to this notion called designated linkability [127] does not allow linkability by default, but instead allows links to be revealed at will by a designated party. Another notion called traceable ring signatures [79] considers a setting where signatures are generated w.r.t. “tags” and each member may sign at most a single message (say, a vote) w.r.t. a particular tag, or else his identity will be revealed. Yet another notion called mesh signatures [43] allows signing with respect to more complex predicates regarding ring membership: i.e., not just disjunctions of signer identities but more general monotone access structures.

Group signatures are a different type of signature which allow signing w.r.t. a set of verification keys and provide anonymity of the signer within that set. This concept differs most strikingly from ring signatures in that there is a central authority that (1) sets up the group (i.e., set of signers) and issues keys to members of the group and (2) has the power to revoke the anonymity of the signer of a signature. Notions such as (un)linkability, mentioned above, have been applied to the group signature setting too. Notably, there has also been proposed a notion of deniable group signatures [106], in which the group manager may issue proofs that a particular group member did not sign a particular signature. This bears a little resemblance to our notion of repudiability in ring signatures; however, the presence of a central authority in the group signature setting means these problems are technically rather disparate. [126] construct lattice-based deniable group signatures; however, their technique for deniability is very different from ours, and relies on zero-knowledge proofs of plaintext
inequality for LWE ciphertexts, which do not suffice in our setting.

Several constructions of ring signatures based on lattice assumptions have been proposed (e.g., [44, 134, 19]). The only other construction of ring signatures based on ZAPs is [33], to our knowledge. Numerous other ring signature constructions have been proposed, mostly based on various assumptions on bilinear maps, many but not all of which are in the random oracle model (e.g., [145, 158, 42]).

Finally, we mention two works in the lattice trapdoor literature: the seminal [3], and the more recent [138]. The latter is newer than [83], whose trapdoors we use in our second construction, carried over from their use in the earlier ring signatures of [44].

6.2 Anonymity and unforgeability of ring signatures

In this section we give standard ring signature definitions: syntax, correctness, anonymity, and unforgeability. Our definitions are presented somewhat differently (for motivations described later) but are equivalent to the correspondingly named definitions of [33].

**Definition 6.2.1** (Ring signature). A ring signature scheme is a triple of PPT algorithms \( RS = (\text{Gen}, \text{Sign}, \text{Verify}) \), satisfying the three properties of correctness (Definition 6.2.2), anonymity (Definitions 6.2.5–6.2.7), and unforgeability (Definition 6.2.9). The syntax of Gen, Sign, and Verify follows.

- \( \text{Gen}(1^k) \) takes as input the security parameter \( k \) and outputs a verification key \( vk \) and a signing key \( sk \).
- \( \text{Sign}(R, sk, m) \) takes as input a signing key \( sk \), a message \( m \), and a set of verification keys \( R = \{vk_1, \ldots, vk_N\} \), and outputs a signature \( \sigma \). The set \( R \) is also known as a “ring.”
- \( \text{Verify}(R, \sigma, m) \) takes as input a set \( R \) of verification keys, a signature \( \sigma \), and a message \( m \), and outputs a single bit indicating whether or not \( \sigma \) is a valid signature on \( m \) w.r.t. \( R \).
Where it may not be clear from context, we sometimes write RS.Gen, RS.Sign, RS.RS to denote the Gen, Sign, Verify algorithms belonging to RS.

**Definition 6.2.2** (Correctness). A ring signature scheme \( RS = (\text{Gen}, \text{Sign}, \text{Verify}) \) satisfies correctness if there is a negligible function \( \varepsilon \) such that for any \( N = \text{poly}(k) \), for any \( N \) key pairs

\[
(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \text{Gen}(1^k)
\]

and any \( i \in [N] \), it holds for any message \( m \) that

\[
\text{Pr}[\text{Verify}(R, \text{Sign}(R, sk_i, m), m) = 1] = 1 - \varepsilon(k),
\]

where \( R = \{vk_1, \ldots, vk_N\} \). We say \( RS \) satisfies perfect correctness if (6.1) holds for \( \varepsilon = 0 \).

### 6.2.1 Anonymity

Anonymity is another desired property. First, we give a framework for anonymity definitions in terms of “oracle sets” — this is a useful abstraction for the purpose of combining multiple definitions such as repudiability and claimability. Then we give two more concrete anonymity definitions using the framework. The definitions given herein are equivalent to the correspondingly named definitions in [33].

**Definition 6.2.3** \(((\mathcal{O}_1, \mathcal{O}_2, \alpha)\text{-anonymity})\). Let \( \mathcal{O}_1, \mathcal{O}_2 \) be sets of oracles, where each oracle in the set is parametrized by a list of key-pairs. Define \( \text{Corr}_{(vk_1, sk_1), \ldots, (vk_N, sk_N)} \) to take as input \( i \in [N] \) and output \( \omega_i \leftarrow \text{Gen}^{-1}(vk_i, sk_i) \).

A ring signature scheme \( RS = (\text{Gen}, \text{Sign}, \text{Verify}) \) satisfies \(((\mathcal{O}_1, \mathcal{O}_2, \alpha)\text{-anonymity})\) if for any PPT adversary \( A \) and any polynomial \( N = \text{poly}(k) \), \( \Pr[b' = b] \) in the above game is negligibly close to 1/2. That is, formally, \( \forall \text{ PPT } A = (A_1, A_2), N = \text{poly}(k) \),

\footnote{Upon the first invocation on an input \( i \), \( \text{Corr} \) samples \( \omega_i \leftarrow \text{Gen}^{-1}(vk_i, sk_i) \), stores it, and outputs it. If \( \text{Corr} \) is queried twice on the same input \( i \) then it outputs the same \( \omega_i \) that was previously stored.}

\[179\]
there is a negligible function $\varepsilon$ such that

$$\Pr \left[ \begin{array}{c}
\left((m^*, i_0^*, i_1^*, R^*), s\right) \leftarrow A_{\mathcal{O}, \mathcal{Corr}}(vk_1, \ldots, vk_N) \\
b \leftarrow \{0, 1\} \\
\sigma \leftarrow \text{Sign}(R^* \cup \{vk_i, sk_i\}, sk_i, m^*) \\
b' \leftarrow A_{\mathcal{O}, \mathcal{Corr}}(s, \sigma)
\end{array} \right] : b' = b \land |\{i_0^*, i_1^*\} \cap I| \leq \alpha < \frac{1}{2} + \varepsilon(k),
$$

(6.2)

where $I$ is the set of queries to the corruption oracle; and the notation $A_{\mathcal{O}, \mathcal{Corr}}$ means that for each oracle $O$ in $\mathcal{O}$, $A$ has oracle access to $O(vk_1, sk_1), \ldots, (vk_N, sk_N)$, and $A$ also has oracle access to $\mathcal{Corr}(vk_1, sk_1), \ldots, (vk_N, sk_N)$. For brevity, we write $(O, \alpha)$-anonymity to denote $(O, \alpha, \alpha)$-anonymity.

**Definition 6.2.4** (Signing oracle $OSign$). For a ring signature scheme $RS$, the oracle $OSign(vk, sk)$ is defined to take as input $i \in [n]$, a message $m$, and a set $R$, and output $RS.Sign(R \cup \{vk_i\}, sk_i, m)$. When the oracle is invoked with respect to a single key pair (i.e., $OSign(vk, sk)$), we treat the oracle as taking only two inputs, $m$ and $R$, since $i$ is superfluous in this case.

**Definition 6.2.5** (Anonymity against adversarially chosen keys). A ring signature scheme $RS = (Gen, Sign, Verify)$ satisfies anonymity against adversarially chosen keys if it is $(\{OSign\}, \emptyset, 0)$-anonymous. Moreover, $RS$ satisfies adaptive anonymity against adversarially chosen keys if it is $(\{OSign\}, 0)$-anonymous.

Definition 6.2.5 captures the guarantee that as long as there are at least two honest parties in a ring (represented by $i_0^*, i_1^*$), even if all other parties in the ring are corrupted by an adversary, the adversary cannot tell which of the honest parties produced a signature. We can also consider an even stronger definition where the adversary may get all but one or even all the secret keys (and key generation randomness) of parties in the ring, and anonymity still holds. This is defined formally below.

**Definition 6.2.6** (Anonymity against attribution attacks). A ring signature scheme $RS = (Gen, Sign, Verify)$ satisfies anonymity against attribution attacks if it is $(\{OSign\}, \emptyset, 1)$-
anonymous. Moreover, RS satisfies adaptive anonymity against attribution attacks if it is \((\{\text{OSign}\}, 1)\)-anonymous.

**Definition 6.2.7** (Anonymity against full key exposure). A ring signature scheme \(RS = (\text{Gen}, \text{Sign}, \text{Verify})\) satisfies anonymity against full key exposure if it is \((\{\text{OSign}\}, \varnothing, 2)\)-anonymous. Moreover, RS satisfies adaptive anonymity against full key exposure if it is \((\{\text{OSign}\}, 2)\)-anonymous.

We remark that the adaptive variants of the above definitions are not presented in [33]. For the rest of the paper, we refer primarily to anonymity against adversarially chosen keys — as mentioned in Section 6.1, this is the notion that is compatible with repudiability. The stronger anonymity notions are implied by unclaimability.

### 6.2.2 Unforgeability

We define a framework for unforgeability definitions, too. Then we give a specific definition using the framework. Though we only instantiate the framework once in this subsection, we will find the framework useful to reference in later sections of this paper.

**Definition 6.2.8** (\(\mathcal{O}\)-unforgeability). Let \(\mathcal{O}\) be a set of oracles, where each oracle in the set is parametrized by a list of key-pairs. A ring signature scheme \(RS = (\text{Gen}, \text{Sign}, \text{Verify})\) is \(\mathcal{O}\)-unforgeable if for any PPT \(A\) and any \(N = \text{poly}(k)\), there is a negligible function \(\varepsilon\) such that

\[
\Pr \left[ \begin{array}{l}
(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \text{Gen}(1^k) \\
(R^*, m^*, \sigma^*) \leftarrow A^{\mathcal{O}, \text{OSign}, \text{Corr}}(vk_1, \ldots, vk_N) \\
b \leftarrow \text{Verify}(R^*, \sigma^*, m^*) 
\end{array} \right. \\
\text{s.t. } \begin{array}{c}
b = 1 \land R^* \subseteq \{vk_1, \ldots, vk_N\} \setminus I \land Q \cap \{(\cdot, m^*, R^*)\} = \varnothing \\
< \varepsilon(k),
\end{array}
\]

where the notation \(A^{\mathcal{O}, \text{OSign}, \text{Corr}}\) is defined as in Definition 6.2.3, and \(I\) and \(Q\) are the sets of queries made to the corruption and signing oracles respectively.

We refer to the event that the conditions on the right-hand side of the colon in the above probability expression are met as a “successful forgery.”
Definition 6.2.9 (Unforgeability of ring signatures). A ring signature scheme $RS = (Gen, Sign, Verify)$ is unforgeable if it is $\emptyset$-unforgeable.\footnote{This is the definition described as unforgeability with respect to insider corruption in [33], and is the strongest of the three unforgeability definitions considered therein.}

6.3 New definitions: repudiability, claimability, and more

6.3.1 Repudiable ring signatures

Repudiability addresses the question of whether or not members of the ring can prove that they did not sign a particular message.

Definition 6.3.1 (Repudiable ring signature). A repudiable ring signature scheme is a ring signature scheme with an additional pair of algorithms $(Repudiate, VerRepud)$, satisfying the four properties of correctness (Definition 6.2.2), repudiability (Definition 6.3.3), anonymity (Definition 6.3.4), and unforgeability (Definition 6.3.5). The syntax of $Repudiate$ and $VerRepud$ follows.

- $Repudiate(R, sk, \sigma)$ takes as input a signing key $sk$, a ring signature $\sigma$, and a set of verification keys $R = \{vk_1, \ldots, vk_N\}$, and outputs a repudiation $\xi$.
- $VerRepud(R, vk, \sigma, \xi)$ takes as input a set $R$ of verification keys, a signature $\sigma$, a repudiation $\xi$, and an identity $vk$, and outputs a single bit indicating whether or not $\xi$ is a valid repudiation of signature $\sigma$ for identity $vk$.

Definition 6.3.2 (Repudiation oracle ORpd). For a repudiable ring signature scheme $RS$, the oracle $ORpd_{(vk_1, sk_1), \ldots, (vk_N, sk_N)}$ is defined to take as input $i \in [n]$, a signature $\sigma$, and a set $R$, and output $RS.Repudiate(R \cup \{vk_i\}, sk_i, \sigma)$. When the oracle is invoked with respect to a single key pair (i.e., $ORpd_{(vk, sk)}$), we treat the oracle as taking only two inputs, $\sigma$ and $R$, since $i$ is superfluous in this case.

Definition 6.3.3 (Repudiation). A ring signature scheme $\Sigma = (Gen, Sign, Verify)$ satisfies repudiability if equipped with algorithms $(Repudiate, VerRepud)$ such that for
any (possibly adversarial) PPT signing algorithm MalSign, there exists a negligible function $\varepsilon$ such that

$$
\begin{align*}
\Pr & \left[ (vk, sk) \leftarrow \text{Gen}(1^k) \\
& (\sigma, m, R') \leftarrow \text{MalSign}^{O,\text{ORpd}(vk, sk)}(vk) \\
& \xi \leftarrow \text{Repudiate}(R', sk, \sigma) \quad : \quad b = 1 \lor b' = 0 \\
& b \leftarrow \text{VerRepud}(R', vk, \sigma, \xi) \\
& b' \leftarrow \text{Verify}(R', \sigma, m) \\
\right]
> 1 - \varepsilon(k)
\end{align*}
$$

(6.3)

where $O = \{\text{OSign}\}$, and also, for any (possibly adversarial) sign-and-repudiate algorithm MalSignRepud, there exists a negligible function $\varepsilon$ such that for any $N = \text{poly}(k)$,

$$
\begin{align*}
\Pr & \left[ (vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \text{Gen}(1^k) \\
& (\sigma, R', m, \{\xi_{vk}\}_{vk \in R \setminus R}) \leftarrow \text{MalSignRepud}^{O}(R) \\
& \forall vk \in R \setminus R, b_{vk} \leftarrow \text{VerRepud}(R', vk, \sigma, \xi_{vk}) \\
& b' \leftarrow \text{Verify}(R', \sigma, m) \\
\right]
> 1 - \varepsilon(k)
\end{align*}
$$

(6.4)

where $R = \{vk_1, \ldots, vk_N\}$, $O = \{\text{OSign, ORpd}\}$, and $Q$ is the set of queries made to oracle $\text{OSign}$.

Note that (6.3) captures the requirement “good people can repudiate,” i.e., that for any (possibly maliciously generated) signature, an honest party who did not produce it should be able to successfully repudiate. (6.4) captures the requirements that “bad people cannot repudiate a signature they produced,” i.e., addressing the case where the malicious signature and repudiation are both produced using the key being verified, and thus we want the signer to be unable to produce a valid repudiation.

**Anonymity and unforgeability of repudiable ring signatures**

The definitions of anonymity and unforgeability need to be adapted for repudiable ring signature schemes, to allow the adversary access to a repudiation oracle as described next.
Definition 6.3.4 (Anonymity of repudiable ring signatures). A repudiable ring signature scheme
\[(\text{Gen}, \text{Sign}, \text{Verify}, (\text{Repudiate}, \text{VerRepud}))\]
satisfies anonymity against
\{adversarially chosen keys, attribution attacks, full key exposure\}
if \((\text{Gen}, \text{Sign}, \text{Verify})\) is \(\{\text{OSign}, \text{ORpd}\}, \emptyset, \alpha\)-anonymous (Definition 6.2.3) for, respectively,
\[\alpha \in \{2, 1, 0\} \, .\]
Moreover, the repudiable ring signature satisfies the adaptive variants of the above anonymity definitions if \((\text{Gen}, \text{Sign}, \text{Verify})\) is \(\{\text{OSign}, \text{ORpd}\}, \alpha\)-anonymous for \(\alpha \in \{2, 1, 0\}\) respectively.

Definition 6.3.5 (Unforgeability of repudiable ring signatures). A repudiable ring signature scheme
\[(\text{Gen}, \text{Sign}, \text{Verify}, (\text{Repudiate}, \text{VerRepud}))\]
is unforgeable if \((\text{Gen}, \text{Sign}, \text{Verify})\) is \(\{\text{ORpd}\}\)-unforgeable (Definition 6.2.8).

6.3.2 Unrepudiable ring signatures

It is also interesting to consider a notion where it is not possible for a party to prove to others that he did not produce that signature.

We omit a formal definition of unrepudiability here, as further discussion and definitions of relevant ideas will be given in Section 6.3.4, regarding the related and stronger notion of unclaimability.

6.3.3 Claimable ring signatures

Another condition we are interested in is whether the actual signer can prove later that they were the signer, without remembering the signing randomness.
Definition 6.3.6 (Claimable ring signature). A claimable ring signature scheme is a ring signature scheme with an additional pair of algorithms (Claim, VerClaim), satisfying the four properties of correctness (Definition 6.2.2), claimability (Definition 6.3.8), anonymity (Definition 6.3.9), and unforgeability (Definition 6.3.10).

The syntax of Claim and VerClaim follows.

- **Claim**\((R, sk, \sigma)\) takes as input a signing key \(sk\), a ring signature \(\sigma\), and a set of verification keys \(R = \{vk_1, \ldots, vk_N\}\), and outputs a claim \(\zeta\).
- **VerClaim**\((R, vk, \sigma, \zeta)\) takes as input a set \(R\) of verification keys, a signature \(\sigma\), a claim \(\zeta\), and an identity \(vk\), and outputs a single bit indicating whether or not \(\zeta\) is a valid claim of signature \(\sigma\) for identity \(vk\).

Definition 6.3.7 (Claim oracle **OClaim**). For a claimable ring signature scheme \(RS\), the oracle **OClaim**\((vk_1, sk_1), \ldots, (vk_N, sk_N)\) is defined to take as input \(i \in [n]\), a set \(R\), and a signature \(\sigma\), and output \(RS.\text{Claim}(R, sk, \sigma)\). When the oracle is invoked with respect to a single key pair (i.e., \(OClaim(vk, sk)\)), we treat the oracle as taking only two inputs, \(R\) and \(\sigma\), since \(i\) is superfluous in this case.

Additionally, we define the oracle **OClaim**\((\sigma^*)\)(\(vk_1, sk_1), \ldots, (vk_N, sk_N)\) to output \(\perp\) when it receives the signature \(\sigma^*\) as input, and otherwise to give the same response as **OClaim**\((vk_1, sk_1), \ldots, (vk_N, sk_N)\).

Definition 6.3.8 (Claimability). A ring signature scheme \((\text{Gen}, \text{Sign}, \text{Verify})\) is claimable if equipped with algorithms (Claim, VerClaim) such that the following conditions hold.

1. (Honest signer can claim) There exists a negligible function \(\varepsilon\) such that for any \(N = \text{poly}(k)\) and \((vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \text{Gen}(1^k)\) and any \(i \in [N]\), it holds for any message \(m\) that

\[
\Pr [\sigma \leftarrow \text{Sign}(R, sk_i, m) : \text{VerClaim}(R, vk_i, \sigma, \text{Claim}(R, sk_i, \sigma)) = 1] > 1 - \varepsilon(k),
\]

where \(R = \{vk_1, \ldots, vk_N\}\).

2. (Non-signers cannot claim) For any (possibly adversarial) PPT sampler \(A\) and...
claiming algorithm \textbf{MalClaim}, there exists a negligible function $\varepsilon$ such that

\[
\begin{array}{l}
\Pr \left[ \begin{array}{l}
(vk, sk) \leftarrow \text{Gen}(1^k) \\
(R', m) \leftarrow A^{O, O\text{Claim}(vk, sk)}(1^k, vk) \\
\sigma \leftarrow \text{Sign}(R' \cup \{vk\}, sk, m) \\
(\zeta, vk') \leftarrow \text{MalClaim}^{O, O\text{Claim}(vk, sk)}(R' \cup \{vk\}, \sigma) \\
b \leftarrow \text{VerClaim}(R' \cup \{vk\}, vk', \sigma, \zeta) \\
b' \leftarrow \text{Verify}(R' \cup \{vk\}, \sigma, m)
\end{array} \right] < \varepsilon(k).
\end{array}
\]

where $O = \{\text{OSign}\}$.

3. (Malicious signer cannot claim on behalf of another party) For any PPT adversary $A$, we have that

\[
\begin{array}{l}
\Pr \left[ \begin{array}{l}
(vk, sk) \leftarrow \text{Gen}(1^k) \\
(R', m, \sigma, \zeta) \leftarrow A^{O, O\text{Claim}(vk, sk)}(1^k, vk) \\
b \leftarrow \text{VerClaim}(R' \cup \{vk\}, vk, \sigma, \zeta) \\
b' \leftarrow \text{Verify}(R' \cup \{vk\}, \sigma, m)
\end{array} \right] < \varepsilon(k).
\end{array}
\]

where $O = \{\text{OSign}\}$ and $Q$ is the set of queries made to oracle $O\text{Claim}_{vk, sk}$.

**Anonymity and unforgeability of claimable ring signatures**

The definitions of anonymity and unforgeability need to be adapted for claimable ring signature schemes, to allow the adversary access to a claim oracle as described next.

**Definition 6.3.9** (Anonymity of claimable ring signatures). A claimable ring signature scheme

\[
(\text{Gen}, \text{Sign}, \text{Verify}, (\text{Claim}, \text{VerClaim}))
\]

satisfies anonymity against

\{adversarially chosen keys, attribution attacks, full key exposure\}

if $(\text{Gen}, \text{Sign}, \text{Verify})$ is $(\{\text{OSign}, O\text{Claim}\}, \varnothing, \alpha)$-anonymous (Definition 6.2.3) for, respectively,

$\alpha \in \{2, 1, 0\}$.
Moreover, the claimable ring signature satisfies the adaptive variants of the above anonymity definitions if \((\text{Gen, Sign, Verify})\) is \((\{\text{OSign, OClaim}\}, \{\text{OSign, OClaim}^{(\sigma)}\}, \alpha)\)-anonymous for \(\alpha \in \{2, 1, 0\}\) respectively, where \(\sigma\) denotes the challenge signature in the anonymity experiment (described in (6.2) of Definition 6.2.3).

**Definition 6.3.10** (Unforgeability of claimable ring signatures). A claimable ring signature scheme

\[(\text{Gen, Sign, Verify, (Claim, VerClaim)})\]

is unforgeable if \((\text{Gen, Sign, Verify})\) is \(\{\text{OClaim}\}\)-unforgeable (Definition 6.2.8).

### 6.3.4 Unclaimable ring signatures

An unclaimable ring signature scheme has the property that the signer cannot later convince anyone of her identity. That is, for any function that the true signer can compute given the signing randomness and the secret key, any other member of the ring can compute an indistinguishable function.

It suffices for any member of the ring to be able to extract signing randomness distributed indistinguishably from true signing randomness, that would produce the given signature under their secret key. More formally, the following guarantee should hold.

**Definition 6.3.11** (Unclaimable ring signatures). A unclaimable ring signature scheme is a ring signature scheme augmented with an additional algorithm \text{ExtractRandomness} as follows.

- \text{ExtractRandomness}(R, sk, \sigma, m) takes as input a ring \(R\), a secret key \(sk\), a signature \(\sigma\) and a message \(m\). If \(sk\) is one of the secret keys for ring \(R\), and \(\sigma\) is a signature for message \(m\) with respect to \(R\), then it outputs randomness \(\rho\).

**Definition 6.3.12.** A ring signature scheme \((\text{Gen, Sign, Verify})\) is (statistically) unclaimable if equipped with an algorithm \text{ExtractRandomness} such that there exists a negligible function \(\epsilon\) such that for any \(N = \text{poly}(k)\), any \(i, i' \in [N]\), and any message \(m\), the following holds. Let \(R\) be the distribution for which the randomness for \(\text{Sign}\) is
sampled. Let \( (v_1, s_1), \ldots, (v_N, s_N) \leftarrow \text{Gen}(1^k) \), let \( R = \{v_1, \ldots, v_N\} \), and \( S = \{s_1, \ldots, s_N\} \). Let \( p \leftarrow \mathcal{R} \), \( \sigma' \leftarrow \text{Sign}(R, s_i, m; p) \), and \( \rho' \leftarrow \text{ExtractRandomness}(R, s_{i'}, \sigma, m) \). Let \( \rho'' \leftarrow \mathcal{R} \), and \( \sigma'' \leftarrow \text{Sign}(R, s_{i''}, m; \rho'') \). Then

\[
(R, S, \rho', \sigma') \approx_i (R, S, \rho'', \sigma'').
\]

**On unrepudiability.** Unrepudiability is implied by unclaimability — or in other words, unclaimability is not compatible with repudiability. The reason for this is that any convincing repudiation procedure can be converted into a claiming procedure, as follows. For a repudiation procedure to be meaningful, it must be infeasible for a valid repudiation to be produced with respect to the true signer's key material. Thus, a convincing *claim* can be made by the true signer by demonstrating that when the repudiation procedure is run with respect to her key material, the procedure fails or outputs an invalid repudiation.

6.3.5 Repudiable-and-claimable ring signatures

Suppose that \( (\text{Gen}, \text{Sign}, \text{Verify}) \) is a ring signature scheme, and there are algorithms \text{Repudiate}, \text{VerRepud}, \text{Claim}, and \text{VerClaim} such that

\[
(\text{Gen}, \text{Sign}, \text{Verify}, (\text{Repudiate}, \text{VerRepud})) \text{ and } (\text{Gen}, \text{Sign}, \text{Verify}, (\text{Claim}, \text{VerClaim}))
\]

are a repudiable ring signature scheme and a claimable ring signature scheme respectively (Definitions 6.3.1 and 6.3.6). The seven algorithms together do *not* necessarily satisfy the natural notion of a "repudiable-and-claimable" ring signature scheme.

The reason, in a nutshell, is that Definition 6.3.1 (repudiability) does *not* allow the adversary a claim oracle, and likewise Definition 6.3.6 (claimability) does *not* allow the adversary a repudiation oracle. Indeed, it would not make sense even syntactically for the "other oracles" to be provided: since each of Definitions 6.3.1 and 6.3.6 is defined with respect to a quintuple of algorithms either containing \text{Repudiate} but not \text{Claim}, or vice versa, the concept of the "other oracle" is undefined within the scope of each
Thus, it could be that when an adversary has access to both a claim and a repudiation oracle, the resulting scheme is no longer secure. Indeed, there are simple (though arguably unnatural) examples of schemes where this happens, such as the following.

**Example 6.3.13.** Given any ring signature scheme, augment the signing key $sk$ to a new signing key $sk' = (sk, \eta_0, \eta_1)$ that additionally contains a pair $\eta_0, \eta_1$ such that $\eta_0$ is sampled uniformly randomly and $\eta_0 \oplus \eta_1 = sk$. Sign works just as in the original scheme, using only $sk$ and ignoring $\eta_0, \eta_1$. Repudiate produces repudiations just as in the original scheme, but additionally appends $\eta_0$ to every repudiation. Claim produces claims just as in the original scheme, but additionally appends $\eta_1$ to every repudiation. This modified scheme would be repudiable if the original scheme was, and also claimable if the original scheme was. However, an adversary that could see both a repudiation and a claim would straightforwardly be able to recover $sk$ and thereby forge signatures.

The natural security definition for a repudiable-and-claimable ring signature scheme is to include both repudiation and claim oracles throughout the repudiability, claimability, anonymity, and unforgeability definitions. The resulting formal definitions are somewhat repetitive and can be skipped without great bearing on the rest of the chapter.

**Definition 6.3.14 (Repudiable-and-claimable ring signature).** A repudiable-and-claimable ring signature scheme is a ring signature scheme with an additional quadruple of algorithms

$$(\text{Repudiate}, \text{VerRepud}, \text{Claim}, \text{VerClaim}) ,$$

satisfying the five properties of correctness (Definition 6.2.2), repudiability (Definition 6.3.15) claimability (Definition 6.3.16), anonymity (Definition 6.3.17), and unforgeability (Definition 6.3.18).

The syntax of (Repudiate, VerRepud) and (Claim, VerClaim) are as defined in Definitions 6.3.3 and 6.3.8 respectively.
Definition 6.3.15 (Repudiability of repudiable-and-claimable ring signatures). A ring signature scheme \( \Sigma = (\text{Gen, Sign, Verify}) \) satisfies repudiability if equipped with algorithms

\[(\text{Repudiate, VerRepud, Claim, VerClaim})\]

such that for any (possibly adversarial) PPT signing algorithm \( \text{MalSign} \), there exists a negligible function \( \epsilon \) such that (6.3) and (6.4) (from Definition 6.3.3) are satisfied when \( \mathcal{O} = \{\text{OSign, ORpd, OClaim}\} \).

Definition 6.3.16 (Claimability of repudiable-and-claimable ring signatures). A ring signature scheme \( \Sigma = (\text{Gen, Sign, Verify}) \) satisfies repudiability if equipped with algorithms

\[(\text{Repudiate, VerRepud, Claim, VerClaim})\]

such that

Definition 6.3.17 (Anonymity of repudiable-and-claimable ring signatures). A repudiable-and-claimable ring signature scheme

\[(\text{Gen, Sign, Verify, (Repudiate, VerRepud, Claim, VerClaim})\]

satisfies anonymity against

\{adversarially chosen keys, attribution attacks, full key exposure\}

if \( (\text{Gen, Sign, Verify}) \) is \( \{\text{OSign, ORpd, OClaim}\}, \emptyset, \alpha \)-anonymous (Definition 6.2.3) for, respectively,

\[\alpha \in \{2, 1, 0\} \]

Moreover, the claimable ring signature satisfies the adaptive variants of the above anonymity definitions if \( (\text{Gen, Sign, Verify}) \) is \( \{\text{OSign, ORpd, OClaim}\}, \alpha \)-anonymous for \( \alpha \in \{2, 1, 0\} \) respectively.

Definition 6.3.18 (Unforgeability of repudiable-and-claimable ring signatures). A
repudiable-and-claimable ring signature scheme

(Gen, Sign, Verify, (Repudiate, VerRepud, Claim, VerClaim))

is unforgeable if (Gen, Sign, Verify) is \{ORpd, OClaim\}-unforgeable (Definition 6.2.8).

6.4 Repudiable construction

6.4.1 Building blocks

ZAPs. ZAPs are two-message public coin witness indistinguishable proofs [68].

Definition 6.4.1 (ZAP). A ZAP for an NP language \(L\) with witness relation \(R_L\) is a triple of algorithms \(\text{ZAP}_L = (\text{ZAP.Setup}_L, \text{ZAP.Prove}_L, \text{ZAP.Verify}_L)\), where \(\text{ZAP.Setup}\) and \(\text{ZAP.Prove}\) are PPT and \(\text{ZAP.Verify}\) is polynomial-time and deterministic, satisfying the following properties.

Public coin. For some polynomial \(\ell = \ell(k)\), \(\text{ZAP.Setup}\) is the algorithm that on input \(1^k\), outputs a uniformly random element of \(\{0, 1\}^\ell\).

Completeness. For \((x, w) \in R_L\) and any \(\rho \in \{0, 1\}^{\ell(k)}\) we have

\[
\Pr_{\pi \leftarrow \text{ZAP.Prove}(\rho, x, w)} [\text{ZAP.Verify}(\rho, \pi, x) = 1] = 1.
\]

Adaptive soundness. There exists a negligible function \(\epsilon\) such that

\[
\Pr_{\rho \leftarrow \text{ZAP.Setup}(1^k)} [\exists (x, \pi) : x \notin L \land \text{ZAP.Verify}(\rho, \pi, x)] \leq \epsilon(k).
\]

Witness indistinguishability. For any sequences \(\{\rho_k\}_{k \in \mathbb{N}}, \{x_k\}_{k \in \mathbb{N}}, \{w_{0,k}\}_{k \in \mathbb{N}}, \{w_{1,k}\}_{k \in \mathbb{N}}\), where for all \(k\), \(\rho_k \in \{0, 1\}^{\ell(k)}\), \(x_k \in L\) and \((x_k, w_{0,k}), (x_k, w_{1,k}) \in R_L\), we have that the following pair of ensembles is computationally indistinguishable:

\[
\{\text{ZAP.Prove}(\rho_k, x_k, w_{0,k})\}_{k \in \mathbb{N}} \approx \{\text{ZAP.Prove}(\rho_k, x_k, w_{1,k})\}_{k \in \mathbb{N}}.
\]
In this work, for simplicity, we will assume use of a ZAP for some NP-complete language $L_{NP}$ (with witness relation $\mathcal{R}_{L_{NP}}$) and for any $L \in NP$ with witness relation $\mathcal{R}_{L}$, we define $\text{ZAP.Prove}_L$ and $\text{ZAP.Verify}_L$ as follows.

- $\text{ZAP.Prove}_L$ takes as input a triple $(\rho, x, w)$. If $(x, w) \notin \mathcal{R}_L$, then output ⊥. Otherwise, use an NP reduction on $(x, w)$ to get a pair $(x_{NP}, w_{NP}) \in \mathcal{R}_{L_{NP}}$, and output $\text{ZAP.Prove}(\rho, x, w)$.
- $\text{ZAP.Verify}_L$ takes as input a triple $(\rho, \pi, x)$, uses the same NP reduction to obtain $x_{NP}$ (which is in $L_{NP}$ iff $x \in L$), and outputs $\text{ZAP.Verify}(\rho, \pi, x)$.

Verifiable random functions (VRFs) are another of our main building blocks [136]. The important property of VRFs that we rely on is residual pseudorandomness, i.e., that VRF outputs on inputs for which the adversary has not received proofs remain indistinguishable from random.

**Definition 6.4.2 (VRF).** A verifiable random function (VRF) is a tuple of algorithms $\text{VRF} = (\text{VRF.Gen}, \text{VRF.Eval}, \text{VRF.Prove}, \text{VRF.Verify})$, where Gen and Verify are PPT and Eval and Prove are polynomial-time and deterministic, satisfying:

**Complete provability** With probability $1 - 2^{-\Omega(k)}$ over $(pk, sk) \leftarrow \text{VRF.Gen}(1^k)$, we have for all inputs $x$ that

$$\Pr[\text{VRF.Verify}(pk, x, \text{VRF.Eval}(sk, x), \text{VRF.Prove}(sk, x)) = 1] > 1 - 2^{-\Omega(k)}$$

**Unique provability** For all $pk, x, y_1, y_2, \tau_1, \tau_2$ with $y_1 \neq y_2$, for either $i = 1$ or $i = 2$ it holds that

$$\Pr[\text{VRF.Verify}(pk, x, y_i, \tau_i) = 1] < 2^{-\Omega(k)}.$$

**Residual pseudorandomness** Let $A = (A_1, A_2)$ be a probabilistic polynomial-time adversary, where both halves of the adversary get oracle access to the VRF evaluation and prove algorithm. Let $(pk, sk) \leftarrow \text{VRF.Gen}(1^k)$, and let

$$(x, s) \leftarrow A_1^{\text{VRF.Eval}(sk, \cdot), \text{VRF.Prove}(sk, \cdot)}(1^k, pk).$$

192
Let \( b \leftarrow \{0, 1\} \), and let \( v \) be either \( \text{VRF.Eval}(sk, x) \) or uniformly random, depending on the choice bit \( b \). Let

\[
b' = A^\text{VRF.Eval}(sk),\text{VRF.Prove}(sk, v, s)\mathbin{[1^k, v, s]}.
\]

Then there is a negligible function \( \varepsilon \) such that

\[
\Pr[b = b' \text{ and } x \notin Q] < \frac{1}{2} + \varepsilon(k),
\]

where \( Q \) is the set of oracle queries made by \( A_1 \) or \( A_2 \) to either oracle.

For simplicity, we assume that \( \text{Eval} \) takes inputs \( x \) of any length, i.e., \( x \in \{0,1\}^* \).

The residual pseudorandomness property of the VRF still holds even if the adversary gets to query many key pairs at once, and gets to adaptively choose to learn some of the secret keys (in this case, residual pseudorandomness should hold of the uncorrupted keys only). This is captured formally in the lemma below.

**Lemma 6.4.3.** Let \( \text{VRF} \) be a a VRF. Then \( \forall \) PPT \( A = (A_1, A_2) \) and all \( N = \text{poly}(k) \), there is a negligible function \( \varepsilon \) such that

\[
\Pr\left[\left(\begin{array}{l}
(pk_1, sk_1), \ldots, (pk_N, sk_N) \leftarrow \text{VRF.Gen}(1^k) \\
(m^*, s) \leftarrow A^{V,\text{Corr}(vk_1, \ldots, vk_N)} \\
\forall i \in [N], y_{i,0} \leftarrow \text{VRF.Eval}(sk_i, m^*) \\
\forall i \in [N], y_{i,1} \leftarrow s \\
b \leftarrow \{0, 1\} \\
b' \leftarrow A_2(s, (y_{i,b})_{i \in [N] \setminus C})
\end{array}\right) : b = b' \land \forall i \in [N] \setminus C, (i, m^*) \notin Q \right] < \frac{1}{2} + \varepsilon(\lambda),
\]

where \( V \) is an oracle that takes as input a pair \((i, m)\) and outputs

\[
(y, \tau) = (\text{VRF.Eval}(sk_i, m), \text{VRF.Prove}(sk_i, m)),
\]

and \( \text{Corr} \) is an oracle that takes as input an index \( i \in [N] \) and outputs \( sk_i \), and \( C \) denotes the set of queries made to the corruption oracle, and \( Q \) denotes the set of the

193
queries to \( V \).

Later in the paper, we refer to the game described by (6.8) as the Parallel VRF Game.

### 6.4.2 Construction

**Construction 6.4.4.** Our construction R-RS is parametrized by ZAP and VRF, where ZAP is a ZAP and VRF is a VRF with input domain \( \{0, 1\}^\ast \) and whose Verify algorithm takes \( \nu \) bits of randomness.

We first present the \( \text{Gen} \) algorithm of our ring signature scheme R-RS.

\[
\text{R-RS.Gen}(1^k)
\]

1. \((vk^1_{VRF}, sk^1_{VRF}), \ldots, (vk^4_{VRF}, sk^4_{VRF}) \leftarrow \text{VRF.Gen}(1^k)\).
   
   Let \( v^k_{VRF} = (vk^1_{VRF}, \ldots, vk^4_{VRF}) \) and \( s^k_{VRF} = (sk^1_{VRF}, \ldots, sk^4_{VRF}) \).
2. \( \rho \leftarrow \text{ZAP.Setup}(1^k) \).
3. \( \bar{\alpha} = (\alpha_1, \ldots, \alpha_M) \leftarrow (\{0, 1\}^\nu)^M \).
4. Output \( vk = (v^k_{VRF}, \rho, \bar{\alpha}) \) and \( sk = (s^k_{VRF}, vk) \).

We now establish some notation before proceeding to the rest of our construction. In the rest of the section, we (implicitly) use the following convention to parse a ring \( R \).

Write \( R = \{vk_1, \ldots, vk_N\} \).

For each \( i \in [N] \), write \( vk_i = (v^i_{VRF} = (vk^{i,1}_{VRF}, \ldots, vk^{i,4}_{VRF}), \rho_i, \bar{\alpha}_i = (\alpha^{i,1}_1, \ldots, \alpha^{i,M}_M)) \).

---

**Definition 6.4.5.** Let \( L \) be the following NP language.

\[
\{(R, m, \varphi, (y_1, y_2, y_3, y_4)) : \exists \iota^*, \iota_1, \iota_2, \iota_3, \iota_4, \iota \text{ s.t. } (b_1 \lor b_2) \land (b_3 \lor b_4) \text{ where } \forall \eta \in \{1, 2, 3, 4\}, b_\eta = \bigwedge_{i \in [N], j \in [M]} \text{VRF.Verify}(v^i_{VRF}k^j_{VRF}, (R, m, \varphi), y_\eta, \iota; \alpha^i_j \oplus \gamma)\}.
\]

---

\(^8\) We include the verification key in \( sk \) so that the \( \text{Sign} \) procedure can identify the verification key in the ring corresponding to the signing key.
We now present the Sign and Verify algorithms of our construction.

R-RS.Sign(\(R, sk, m\))

1. Parse \(R\) as described above and \(sk = ((sk^1_{VRF}, \ldots, sk^4_{VRF}), vk)\).
2. If \(vk \notin R\) output \(\bot\) and halt.
3. Define \(i^* \in [N]\) such that \(vk_{i^*} = vk\).
4. \(\gamma \leftarrow \{0, 1\}^\nu\). (This is used as part of the ZAP witness in Step 6.)
5. \(\varphi \leftarrow \{0, 1\}^\lambda\). (This is used as a salt for the VRF input in Step 7, and output in Step 8.)
6. For \(\eta \in \{1, 2, 3, 4\}\), let
   \[
   y_\eta = \text{VRF.Eval}(sk^\eta_{VRF}, (R, m, \varphi)) \quad \text{and} \quad \tau_\eta = \text{VRF.Prove}(sk^\eta_{VRF}, (R, m, \varphi)) .
   \]
   Let \(\bar{y} = (y_1, \ldots, y_4)\).
7. For each \(i \in [N]\), let \(\pi_i \leftarrow \text{ZAP.Prove}_L(\rho_i, (R, m, \varphi, \bar{y}), (i^*, \tau_1, \bot, \tau_3, \bot, \gamma))\).
   Let \(\bar{\pi} = (\pi_1, \ldots, \pi_N)\).
8. Output \(\sigma = (\bar{\pi}, \bar{y}, \varphi)\).

R-RS.Verify(\(R, \sigma, m\))

1. Parse \(R\) as above and \(\sigma = ((\pi_1, \ldots, \pi_N), \bar{y}, \varphi)\).
2. Output \(\bigwedge_{i \in [N]} \text{ZAP.Verify}_L(\rho_i, \pi_i, (R, m, \varphi, \bar{y}))\).

Now that we have described the main algorithms of R-RS, we proceed to describe the repudiation algorithms for R-RS.

Definition 6.4.6. Let \(L'\) be the following NP language:

\[
\left\{ (R, m, \varphi, (y_1, \ldots, y_4), vk = (pk_E, \bar{vk}_{VRF}, \rho, \bar{\alpha})) : \exists i^*, y'_1, \ldots, y'_4, \tau'_1, \ldots, \tau'_4, \gamma \text{ s.t.} \right. \\
((b'_1 \land b'_2) \lor (b'_3 \land b'_4)) \land vk = vk_{i^*}, \text{ where } \forall \eta \in \{1, 2, 3, 4\}, \\
b'_\eta = \left( y'_\eta \neq y_\eta \land \bigwedge_{i \in [N], j \in [M]} \text{VRF.Verify}(vk^{i*\eta}_{VRF}, (R, m, \varphi), y'_\eta, \tau'_\eta, \alpha'_j \oplus \gamma) \right) \left. \right\}.
\]
R-RS.Repudiate($R, sk, \sigma$)

1. Parse $R$ as above, $sk = (sk^1_{\text{VRF}}, \ldots, sk^4_{\text{VRF}}, vk)$, and $\sigma = (\bar{\pi}, \bar{y}, \phi)$.
2. If $vk \notin R$ output $\bot$ and halt.
3. Define $i^* \in [N]$ such that $vk_{i^*} = vk$.
4. For $\eta \in \{1, 2\}$:
   
   $y'_\eta = \text{VRF.Eval}(sk_{\text{VRF}}^\eta, (R, m, \varphi))$ and $\tau'_\eta = \text{VRF.Prove}(sk_{\text{VRF}}^\eta, (R, m, \varphi))$.

5. $\gamma \leftarrow \{0, 1\}^\nu$. (This is used as part of the ZAP witness in Step 6.)
6. For each $i \in [N]$, let
   
   $\xi_i \leftarrow \text{ZAP.Prove}_L(\rho_i, (R, m, \varphi, \bar{y}, vk), (i^*, y'_1, y'_2, \bot, \bot, \tau'_1, \tau'_2, \bot, \bot, \gamma))$.

7. Output $\xi = (\xi_1, \ldots, \xi_N)$.

R-RS.VerRepud($R, vk, \sigma, \xi$)

1. Parse $R$ as above. If $vk \notin R$, output 1 and halt.
2. Parse $\sigma = (\bar{\pi}, \bar{y}, \phi)$, and $\xi = (\xi_1, \ldots, \xi_N)$.
3. Output $\bigwedge_{i \in [N]} \text{ZAP.Verify}_L(\rho_i, \xi_i, (R, m, \varphi, \bar{y}, vk))$.

Remark 6.4.7. As written, the size of the input $(R, m, \varphi)$ to the VRF scales with the size of the ring $R$, and we have assumed that the VRF has input domain $\{0, 1\}^\ast$, i.e., can take variable-length inputs. When this is not the case, or when it is desirable for efficiency reasons to evaluate the VRF on a smaller input, the scheme can be straightforwardly modified by employing a collision-resistant hash function, and evaluating the VRF on the hash of $(R, m, \varphi)$ rather than on $(R, m, \varphi)$ directly. We have presented the version of the scheme without the hash function, for simplicity of exposition.

6.4.3 Security proof

Theorem 6.4.8. R-RS is a repudiable ring signature scheme.
Proof. Follows directly from Lemmata 6.4.9 (correctness), 6.4.12 (repudiability), 6.4.13 (unforgeability), and 6.4.14 (anonymity).

Lemma 6.4.9 (Correctness of R-RS). R-RS satisfies correctness (Definition 6.2.2).

Correctness is immediate so we omit the proof.

Before presenting the proof of repudiability, we establish the following supporting lemma and corollary.

Lemma 6.4.10. Let \( V \) be a randomized algorithm takes \( \nu \) bits of randomness and outputs one bit. Let \( \beta \in \{0, 1\} \) be a bit, and let \( x \) be an input such that for some negligible \( \varepsilon \),

\[
\Pr [V(1^\lambda, x) = \beta] \geq 1 - \varepsilon. \tag{6.10}
\]

Let \( M \) be a polynomial such that \( M \geq (\nu + \lambda)/\log_2(1/\varepsilon) \). (Note that the right-hand side is at most polynomial since the numerator is polynomial and the denominator is super-constant.) Then the following probability is overwhelming:

\[
\Pr_{(\alpha_1, \ldots, \alpha_M) \sim (\{0, 1\}^\nu)^M} [\forall \gamma \in \{0, 1\}^\nu, \exists i \in [M] \text{ s.t. } V(1^\lambda, x; \alpha_i \oplus \gamma) = \beta]. \tag{6.11}
\]

Proof. Fix any \( \gamma \in \{0, 1\}^\nu \). Let \( \psi_{i, \gamma} = \alpha_i \oplus \gamma \) for each \( i \in [M] \). Since the \( \alpha_i \) are distributed randomly and independently, the distribution of \( (\psi_{i, \gamma})_{i \in [M]} \) is uniform over \( (\{0, 1\}^\nu)^M \) even when conditioned on \( \gamma \). Therefore, conditioned on any given \( \gamma \),

\[
\Pr [\forall i \in [M], V(1^\lambda, x; \psi_{i, \gamma}) \neq \beta] < \varepsilon^M.
\]

There are \( 2^\nu \) possible values of \( \gamma \), so by a union bound,

\[
\Pr [\exists \gamma \in \{0, 1\}^\nu \text{ s.t. } \forall i \in [M], V(1^\lambda, x; \psi_{i, \gamma}) \neq \beta] < 2^\nu \cdot \varepsilon^M.
\]

Since \( M \geq (\nu + \lambda)/\log_2(1/\varepsilon) = \log_2(2^{-\nu - \lambda}) \) by assumption, the right-hand side is at most \( 2^{-\lambda} \), which is negligible. The lemma follows.

The next corollary states that the implication established in Lemma 6.4.10 in fact goes both ways.
Corollary 6.4.11. Let \( \mathcal{V} \) be a randomized algorithm takes \( \nu \) bits of randomness and outputs one bit. Let \( \beta \in \{0, 1\} \) be a bit, and let \( \varepsilon \) be a negligible function. Let \( M \) be a polynomial such that \( M \geq (\nu + \lambda)/\log_2(1/\varepsilon) \). Then (6.10) holds if and only if (6.11) is overwhelming.

Proof. Follows from applying Lemma 6.4.10 for both \( \beta = 0 \) and \( \beta = 1 \).

Next, we give the proof of repudiability of R-RS.

Lemma 6.4.12 (Repudiability of R-RS). R-RS is repudiable (Definition 6.3.3).

Proof. Suppose, for contradiction, that R-RS is not repudiable. Then by Definition 6.3.3, it must be that either (6.3) or (6.4) does not hold. We consider these two possibilities in turn.

Suppose first that (6.3) does not hold for R-RS. Then there is a PPT MalSign that generates a valid signature \( \sigma \) with respect to some ring \( R \), so that \( \sigma \) is not repudiable by some honest party in the ring. That is, the following probability is non-negligible:

\[
\Pr \left[ \begin{array}{c}
(vk, sk) \leftarrow \text{Gen}(1^k) \\
(\sigma, m, R) \leftarrow \text{MalSign}^O(vk) \\
\xi \leftarrow \text{Repudiate}(R, sk, \sigma) \\
b \leftarrow \text{VerRepud}(R, vk, \sigma, \xi) \\
b' \leftarrow \text{Verify}(R, \sigma, m)
\end{array} \right| b = 0 \land b' = 1 \\
\land Q \cap \{\langle \cdot, m, R' \rangle \} = \emptyset,
\]

where \( Q \) is the set of queries to OSign.

Based on MalSign, we build an adversary \( A \) for Parallel VRF Game (defined in Lemma 6.4.3), as follows. A first invokes R-RS.Gen(1^k) to obtain \((vk, sk)\). The \( vk \) is a tuple \((pk_E, \overline{vk}_{VRF}, \rho, \alpha)\), where \( \overline{vk}_{VRF} \) can be parsed further as \((vk_{VRF}^1, \ldots, vk_{VRF}^4)\).

A obtains two verification keys \( vk_{VRF}^3, vk_{VRF}^4 \) from the VRF challenger, and replaces \( vk_{VRF}^3 \) and \( vk_{VRF}^4 \) with these keys, setting

\[ vk' = (pk_E, (vk_{VRF}^1, vk_{VRF}^2, vk_{VRF}^3, vk_{VRF}^4), \rho, \alpha). \]
Let $sk^3_{VRF}, sk^4_{VRF}$ be the VRF secret keys corresponding to $vk^3_{VRF}, vk^4_{VRF}$, respectively. 

Next, $A$ runs $\text{MalSign}(vk')$, answering $\text{MalSign}$'s oracle queries as follows.

- On query $(m'', R'')$ to $O\text{Sign}$: $A$ runs the honest signing algorithm $R-RS.\text{Sign}$ on input $(R'' \cup \{vk'\}, sk, m'')$, with the following modification: in step 6, instead of using $sk^3_{VRF}$ and $sk^4_{VRF}$ to generate $y_3, \tau_3$, and $y_4, \tau_4$, $A$ invokes its VRF oracle.
- On query $(\sigma'', R'')$ to $O\text{Repd}$: $A$ runs the honest repudiation algorithm $R-RS.\text{Repudiate}$ on input $(R'' \cup \{vk'\}, sk, \sigma'')$. (Note that $sk^3_{VRF}$ and $sk^4_{VRF}$ are not used in algorithm $R-RS.\text{Repudiate}$, so we don't need to invoke the VRF oracle here.)

Let $(\sigma, m, R)$ be the output of $\text{MalSign}$. $A$ parses $\sigma = (\bar{\pi}, \bar{y}, \varphi)$ and $\bar{\gamma} = (y_1, \ldots, y_4)$. Then $A$ sends $(R, m, \varphi)$ to the VRF challenger, receiving responses $y'_3$ and $y'_4$. If $y_3 = y'_3$ or $y_4 = y'_4$, $A$ outputs 0. Otherwise, $A$ outputs a random bit. Let us now consider $A$'s behavior in the two cases where the VRF challenger's bit $b$ is equal to 0 and equal to 1.

Case $b = 0$ in Parallel VRF Game. In this case, the view of $\text{MalSign}$ is identical to the view in (6.12), so by assumption $\text{MalSign}$ will win the game — i.e., produce a signature that verifies but is not repudiable by an honest party — with non-negligible probability. Note that whenever $\text{MalSign}$ wins the game, the condition $Q \cap \{ (\cdot, m, R') \} \neq \emptyset$ in (6.12) implies that $A$ has not previously made an oracle query on the VRF challenge message $(R, m, \varphi)$ during the query phase. Let us suppose that $\text{MalSign}$ wins the game described in (6.12), and consider the implications.

By definition, if $R-RS.\text{VerRepud}$ rejects (with non-negligible probability) on an honestly generated repudiation $\xi = (\xi_i)_{i \in [R]}$ generated with respect to $vk'$, then

$$\exists i \in [R] \text{ s.t. } ZAP.\text{Verify}_{L'}(\rho_i, \xi_i, (R, m, \varphi, \bar{y}, vk')) = 0.$$  \hfill (6.13)

By the perfect completeness of the ZAP, and since $\xi$ is honestly generated, (6.13)

---

9Note that $sk^3_{VRF}, sk^4_{VRF}$ are generated by the VRF challenger and not accessible by $A$. 

199
implies that \((R, m, \varphi, \bar{y}, vk') \notin L'\). Moreover, by definition of \(L'\),
\[(R, m, \varphi, \bar{y}, vk') \notin L' \Rightarrow (-b'_3 \lor -b'_4) , \tag{6.14}\]
where \(b'_3, b'_4\) are as defined in Definition 6.4.6. Expanding the definition of \(b'_3, b'_4\), and again using that \(\xi\) is honestly generated, we have that the right-hand side of (6.14) implies:
\[\exists \eta \in \{3, 4\}, i' \in [[R]], j' \in [M], \gamma \text{ s.t.} \]
\[VRF.\text{Verify}(vk'_{\text{VRF}}, (R, m, \varphi), y_{\eta}, VRF.\text{Prove}(sk'_{\text{VRF}}, (R, m, \varphi)); \alpha'_{j'} \oplus \gamma) = 0 . \tag{6.15}\]

Since \(vk' \in R\) is honestly generated by assumption, we have that the \(\bar{a} = (\alpha_1, \ldots, \alpha_M)\) within \(vk'\) is distributed uniformly over \(\{0, 1\}^M\). Then applying Corollary 6.4.11 (setting the algorithm \(\mathcal{V}\) to be \(VRF.\text{Verify}\)): (6.15) implies either
\[\exists \eta \in \{3, 4\} \text{ and } \tau \text{ s.t.} \]
\[\Pr[VRF.\text{Verify}(vk'_{\text{VRF}}, (R, m, \varphi), y_{\eta}, \tau) = 1] \text{ is overwhelming,} \tag{6.16}\]
or a negligible probability event occurred. By the complete and unique provability of the VRF, (6.16) implies that
\[y_3 = VRF.\text{Eval}(sk'_{\text{VRF}}, (R, m, \varphi)) \text{ or } y_4 = VRF.\text{Eval}(sk'_{\text{VRF}}, (R, m, \varphi)) . \tag{6.17}\]

Chaining together the implications, we conclude that (6.17) holds with all but negligible probability conditioned on the non-negligible-probability event of \text{MalSign} winning the game described in (6.12).

Finally, by definition of Parallel VRF Game, when \(b = 0\),
\[y'_3 = VRF.\text{Eval}(sk'_{\text{VRF}}, (R, m, \varphi)) \text{ and } y'_4 = VRF.\text{Eval}(sk'_{\text{VRF}}, (R, m, \varphi)) . \tag{6.18}\]
From (6.17) and (6.18): when \( b = 0 \), there is a non-negligible probability that

\[
y_3 = y'_3 \text{ or } y_4 = y'_4. \tag{6.19}
\]

Recall that (6.19) is the trigger condition for \( A \) to output 0. Therefore, when \( b = 0 \), \( A \) outputs 0 with non-negligible probability (and outputs a random bit the rest of the time).

**Case \( b = 1 \) in Parallel VRF Game.** In this case, \( y'_3 \) and \( y'_4 \) are uniformly random and independent of the rest of the experiment, so they will be distinct from \( y_3 \) and \( y_4 \) with overwhelming probability. Consequently, in this case, with all but negligible probability \( A \) outputs a random bit.

Thus, \( A \) wins Parallel VRF Game with non-negligible probability. This contradicts the security of the VRF. Therefore, R-RS satisfies (6.3).

It remains to show that R-RS satisfies (6.4). Suppose not, for the sake of contradiction. Then there is a PPT algorithm \( \text{MalSignRepud} \) such that the following probability is non-negligible:

\[
\Pr \left[ (v_{k_1}, s_{k_1}), \ldots, (v_{k_N}, s_{k_N}) \leftarrow \text{Gen}(1^k) \right. \right.
\]
\[
(\sigma, R', m, \{\xi_{vk}\}_{vk \in R' \setminus R}) \leftarrow \text{MalSignRepud}^{\mathcal{O}}(R) \quad R' \cap R \neq \emptyset \land \bigwedge_{vk \in R' \setminus R} b_{vk} = 1 \land b' = 1 \land Q \cap \{(\cdot, m, R')\} = \emptyset
\]
\[
\forall vk \in R' \setminus R, \ b_{vk} \leftarrow \text{VerRepud}(R', vk, \sigma, \xi_{vk})
\]
\[
b' \leftarrow \text{Verify}(R', \sigma, m)
\]

where \( R = \{v_{k_1}, \ldots, v_{k_N}\} \), \( \mathcal{O} = \{\text{OSign, ORpd}\} \), and \( Q \) is the set of queries made to oracle \( \text{OSign} \).

Based on \( \text{MalSignRepud} \), we construct another adversary \( A' \) to Parallel VRF Game for \( 2N \) keys, as follows. \( A' \) first samples

\[
(v_{k_1}, s_{k_1}), \ldots, (v_{k_N}, s_{k_N}) \leftarrow \text{Gen}(1^k)
\]
and parses each $v_{ki}$ as

$$v_{ki} = (pk_i^E, v_{k_{VRF}^{i^1}}, v_{k_{VRF}^{i^2}}, v_{k_{VRF}^{i^3^*}}, v_{k_{VRF}^{i^4^*}}, \rho_i, \alpha_i).$$

Next, $A'$ obtains $2N$ verification keys from the VRF challenger. For notational convenience in the rest of the proof, let these $2N$ keys be denoted by $\{v_{ki^{i^3^*^*}}, v_{ki^{i^4^*^*}}\}_{i \in [N]}$. Let the corresponding $2N$ secret keys be denoted by $\{s_{k_{VRF}^{i^3^*}}, s_{k_{VRF}^{i^4^*^*}}\}_{i \in [N]}$.  

Define $v_{ki^*}$ as

$$v_{ki^*} = (pk_i^E, v_{k_{VRF}^{i^1}}, v_{k_{VRF}^{i^2}}, v_{k_{VRF}^{i^3^*}}, v_{k_{VRF}^{i^4^*^*}}, \rho_i, \alpha_i).$$

That is, $v_{ki^*}$ is identical to $v_{ki}$ except that $v_{ki^{i^3^*}}$ and $v_{ki^{i^4^*^*}}$ are replaced by $v_{k_{VRF}^{i^3^*}}, v_{k_{VRF}^{i^4^*^*}}$.

Let $R^* = \{v_{k^*_1}, \ldots, v_{k^*_N}\}$. $A'$ then runs MalSignRepud on input $R^*$, answering its oracle queries as follows.

- On query $(i^"", m^"", R^"")$ to OSign: $A'$ runs the honest signing algorithm R-RS.Sign on input $(R^"" \cup \{v_{ki^*}\}, sk_{i^"", m^""})$, with the following modification: in step 6, instead of using $sk_{i^""}$ to generate $y_3, \tau_3$, and $y_4, \tau_4$, $A'$ invokes the corresponding VRF oracle for keys $v_{k_{VRF}^{i^"^1}}, v_{k_{VRF}^{i^"^2}}, v_{k_{VRF}^{i^"^3^*}}, v_{k_{VRF}^{i^"^4^*^*}}$.
- On query $(i^"", a^"", R^"")$ to ORpd: $A$ runs the honest repudiation algorithm R-RS.Repudiate on input $(R^"" \cup \{v_{ki^*}\}, sk_{i^"", a^""})$. (As noted above, $sk_{i^3}^{VRF}$ and $sk_{i^4}^{VRF}$ are not used in algorithm R-RS.Repudiate, so $A$ does not need to invoke the VRF oracle here.)

Let $(\sigma, \xi, m, R')$ be the output of MalSignRepud. $A'$ parses $\sigma = (\bar{\pi}, \bar{\gamma}, \varphi)$ and $\bar{\gamma} = (y_1, \ldots, y_4)$, then submits $(R', m, \varphi)$ to the VRF challenger and, for each $i \in [N]$, receives responses $y_{3,i}, y_{4,i}$ corresponding to . If there exists any index $i \in [N]$ such that $y_{3,i} = y_3$ or $y_{4,i} = y_4$, $A'$ outputs 0. Otherwise, $A'$ outputs a random bit. Let us now consider the behavior of $A'$ in the two cases where the VRF challenger's bit $b$ is equal to 0 and equal to 1.

\footnote{Note that $\{s_{k_{VRF}^{i^3^*}}, s_{k_{VRF}^{i^4^*^*}}\}_{i \in [N]}$ are generated by the VRF challenger and not accessible by $A'$.}
Case $b = 0$. In this case, the view of MalSignRepud is identical to the view in (6.20), so by assumption MalSignRepud wins the game described in (6.20) with non-negligible probability. Note that whenever MalSignRepud wins the game, the condition $Q \cap \{ (\cdot, m, R') \} \neq \emptyset$ in (6.20) implies that A has not previously made an oracle query on the VRF challenge message $(R', m, \varphi)$ during the query phase. Let us suppose that MalSignRepud wins the game described in (6.20) and consider the implications.

By definition, if $R\text{-RS.Verify}$ accepts (with non-negligible probability) on input $(R', \sigma, m)$, then

$$\forall i \in [d(R')], \ ZAP_\text{Verify}_L(\rho_i, \pi_i, (R', m, \varphi, \vec{y})) = 1. \quad (6.21)$$

$R' \cap R \neq \emptyset$ from our assumption that MalSignRepud wins the game described in (6.20), and therefore we have that at least one $\rho_i$ corresponding to some $vk_i \in R' \cap R$ is honestly generated. Thus, the soundness of the ZAP holds w.r.t. this $\rho_i$, and (6.21) implies that $(R', m, \varphi, \vec{y}) \in L$. Moreover, by the definition of $L$,

$$(R', m, \varphi, \vec{y}) \in L \Rightarrow (b_3 \lor b_4), \quad (6.22)$$

where $b_3, b_4$ are as defined in Definition 6.4.5. Expanding the definitions of $b_3, b_4$, we have that the right-hand side of (6.22) implies:

$$\exists \eta \in \{3, 4\}, i^* \in [d(R')], \tau_1, \ldots, \tau_4, \gamma \text{ s.t. } \forall i' \in [d(R')], j' \in [M],
\text{VRF} \text{-Verify}(vk_{i^*}^{\eta^*}, (R', m, \varphi), y_\eta, \tau; \alpha_{j'}^i \oplus \gamma) = 1. \quad (6.23)$$

Then applying Corollary 6.4.11 (again setting the algorithm $V$ to be VRF.Verify): (6.23) implies either

$$\exists \eta \in \{3, 4\} \text{ and } \tau \text{ s.t.}
\text{Pr} \left[ \text{VRF} \text{-Verify}(vk_{i^*}^{\eta^*}, (R', m, \varphi), y_\eta, \tau) = 1 \right] \text{ is overwhelming}, \quad (6.24)$$

or a negligible probability event occurred. By the complete and unique provability
of the VRF, (6.24) implies that

\[ y_3 = \text{VRF.Eval}(sk_{\text{VRF}}^{3,*}, (R', m, \varphi)) \text{ or } y_4 = \text{VRF.Eval}(sk_{\text{VRF}}^{4,*}, (R', m, \varphi)) . \]  (6.25)

Chaining together the implications, we obtain that (6.25) holds with all but negligible probability conditioned on the non-negligible-probability event of MalSignRepud winning the game described in (6.20).

Finally, by definition of Parallel VRF Game, when \( b = 0 \), for all \( i \in [N] \),

\[ y_{3,i} = \text{VRF.Eval}(sk_{\text{VRF}}^{3,*}, (R', m, \varphi)) \text{ and } y_{4,i} = \text{VRF.Eval}(sk_{\text{VRF}}^{4,*}, (R', m, \varphi)) . \]

It follows that conditioned on \( b = 0 \), there is a non-negligible probability that

\[ \exists i^* \in [N] \text{ s.t. } y_3 = y_{3,i^*} \text{ or } y_4 = y_{4,i^*} . \]  (6.26)

Recall that (6.26) is the trigger condition for \( A' \) to output 0. Therefore, when \( b = 0 \), \( A' \) outputs 0 with non-negligible probability (and outputs a random bit the rest of the time).

**Case** \( b = 1 \). In this case, \( y_3 \) and \( y_4' \) are uniformly random and independent of the rest of the experiment, so with overwhelming probability, they will be distinct from \( y_{3,i} \) and \( y_{4,i} \) for every \( i \in [N] \). Consequently, in this case, with all but negligible probability \( A \) outputs a random bit.

Thus, \( A' \) wins Parallel VRF Game with non-negligible probability. This contradicts the security of the VRF. We therefore conclude that R-RS satisfies (6.4). The lemma follows. \( \square \)

**Lemma 6.4.13** (Unforgeability of R-RS). R-RS is unforgeable (in the sense of Definition 6.3.5).

**Proof.** This proof is very similar to the second half of the proof of Lemma 6.4.12.

Suppose that this is not the case. Then there exists some \( N = \text{poly}(k) \) and some PPT B such that the following probability is non-negligible, where \( I \) and \( Q \) are the
sets of queries made to the corruption and signing oracles respectively:

\[
\begin{align*}
(\nu_{k1}, s_{k1}), \ldots, (\nu_{kN}, s_{kN}) & \leftarrow \text{Gen}(1^k) \\
(\hat{R}^*, m^*, \sigma^*) & \leftarrow \text{B}^{\text{OSign,ORpd,Corr}}(\nu_{k1}, \ldots, \nu_{kN}) \\
b & = 1 \land \hat{R}^* \subseteq \{\nu_{k1}, \ldots, \nu_{kN}\} \setminus I \\
\land Q \cap \{\cdot, m^*, \hat{R}^*\} & = \emptyset
\end{align*}
\]

(6.27)

We build an adversary \( A \) to the parallel VRF security game. \( A \) first samples

\[
(\nu_{k1}, s_{k1}), \ldots, (\nu_{kN}, s_{kN}) \leftarrow \text{Gen}(1^k)
\]

and parses each \( \nu_{ki} \) and \( s_{ki} \) as follows:

\[
\nu_{ki} = (p_{E}^{i}, \nu_{VRF}^{i} = (\nu_{k1}^{i,1}, \nu_{k2}^{i,2}, \nu_{kVRF}^{i,3}, \nu_{k3}^{i,4}), \rho_{i}, \alpha_{i})
\]

\[
s_{ki} = (s_{VRF}^{i} = (s_{k1}^{i,1}, s_{k2}^{i,2}, s_{k3}^{i,3}, s_{k4}^{i,4}, \nu_{ki})
\]

Then \( A \) chooses a random \( i^{*} \leftarrow [N] \), obtains a pair of verification keys \( \nu_{k*}^{3}, \nu_{k*}^{4} \) from the VRF challenger, and lets

\[
\nu_{k*} = (p_{E}^{i}, \nu_{VRF}^{i} = (\nu_{k1}^{*}, \nu_{k2}^{*}, \nu_{k3}^{*}, \nu_{k4}^{*}), \rho_{i}, \alpha_{i}).
\]

Let \( s_{k*}^{3}, s_{k*}^{4} \) denote the VRF secret keys corresponding to \( \nu_{k*}^{3}, \nu_{k*}^{4} \), respectively.

Let \( \nu_{ki}^{*} = \nu_{k*} \) and let \( \nu_{ki}^{*} = \nu_{ki} \) for every \( i \neq i^{*} \). Let \( \hat{R}^* = \{\nu_{k1}^{*}, \ldots, \nu_{kN}^{*}\} \). \( A \) then runs \( B \) on input \( \hat{R}^* \), answering \( B \)'s oracle queries as follows.

- On query \((i'', m'', \hat{R}'')\) to \( \text{OSign} \): \( A \) runs the honest signing algorithm \( \text{R-RS.Sign} \) on input \((\hat{R}'' \cup \{\nu_{k*}^{*}\}, s_{k'''}, m''\), with the following modification if \( i = i^{*} \): in step 6, instead of using \( s_{k*}^{3} \) and \( s_{k*}^{4} \) to generate \( y_3, \tau_3, \) and \( y_4, \tau_4 \), \( A \) invokes its VRF oracle.
- On query \((i'', \sigma'', \hat{R}'')\) to \( \text{ORpd} \): \( A \) runs the honest repudiation algorithm \( \text{R-RS.Repudiate} \) on input \((\hat{R}'' \cup \{\nu_{k*}^{*}\}, s_{k''}, \sigma''\). (\( s_{k*}^{3} \) and \( s_{k*}^{4} \) are not used by \( \text{R-RS.Repudiate} \), so \( A \) does not need to invoke the VRF oracle here.)
• On each invocation of Corr on some index $i \in [N]$: if $i = i^*$ then $A$ outputs a random bit and aborts; otherwise, $A$ responds to $B$ with $s k_i$.

Let $(R', m, \sigma)$ be the output of $B$. $A$ parses $\sigma = (\vec{\pi}, \vec{y}, \varphi)$ and $\vec{y} = (y_1, \ldots, y_4)$, and submits $(R', m, \varphi)$ to the VRF challenger and then receive responses $y'_3$ and $y'_4$. If $y'_3 = y_3$ or $y'_4 = y_4$, $A$ outputs 0. Else, $A$ outputs a random bit.

It remains to show that $A$ distinguishes with non-negligible advantage, in the parallel VRF security game, between VRF outputs and random values.

Let us consider the behavior of $A$ in the two cases where the VRF challenger's bit is equal to 0 and equal to 1.

**Case $b = 0$ in Parallel VRF Game.** In this case, the response that $A$ receives to the challenge $(R', m)$ consists of VRF outputs on input $(R', m)$ with respect to the keys $vk^*_3, vk^*_4 \mathcal{VRF}$. In particular, whenever $B$ does not query Corr on $i^*$, the view of $B$ is identical to the view in (6.27). So, conditioned on $B$ not corrupting $i^*$, $B$ will win the game — i.e., produce a valid signature — with non-negligible probability, by assumption.

Let us consider the probability that $B$ queries Corr for input $i^*$. Recall that this event causes $A$ to abort and output a random bit. The distribution of the view (i.e., verification keys and oracle responses) of $B$ is unaffected by $A$'s choice of $i^*$, until the point at which $B$ submits an oracle query to Corr for input $i^*$ (if at all). The condition $R^* \subseteq \{vk_1, \ldots, vk_N\} \setminus I$ in (6.27) ensures that if $B$ wins the game with non-negligible probability, then $B$ leaves one or more keys uncorrupted with at least that non-negligible probability. Since $i^*$ is chosen at random by $A$, it follows that $Pr[i^* \notin I]$ is non-negligible.

Let $E'$ denote the event that $A$ does not abort (i.e., $i^* \notin I$) and $B$'s output signature verifies (i.e., $R$-$RS$.Verify$(R', \sigma, m) = 1$). We have established that $E'$ occurs with non-negligible probability. Then, by the same argument given from (6.21) to
(6.25) in the proof of Lemma 6.4.12, we have that either

$$\exists j^* \in |R'|, \eta \in \{3, 4\}, \text{ and } \tau \text{ s.t.}$$

$$\Pr \left[ \text{VRF.Verify}(v_{k^{j^*}_\eta}, (R', m, \varphi), y, \tau) = 1 \right] \text{ is overwhelming,}$$

or a negligible probability event occurred. When $j^* = i^*$, this moreover implies

$$y_3 = \text{VRF.Eval}(sk_{i*}^3, (R', m, \varphi)) \text{ or } y_4 = \text{VRF.Eval}(sk_{i*}^3, (R', m, \varphi)) \quad (6.28)$$

Let us consider the probability that $j^* = i^*$. As also observed above, the distribution of the view (i.e., verification keys and oracle responses) of B is unaffected by A's choice of $i^*$, until the point at which B submits an oracle query to Corr for input $i^*$ (if at all). Since $i^*$ is chosen at random by A (and is thus independent of $j^*$), and $i^*, j^* \in |R'|$, $\Pr[i^* = j^*]$ must be non-negligible. Therefore, (6.28) holds with non-negligible probability.

Finally, by definition of the , whenever the challenger's bit $b$ is 0,

$$y'_3 = \text{VRF.Eval}(sk_{i*}^3, (R', m, \varphi)) \text{ or } y'_4 = \text{VRF.Eval}(sk_{i*}^3, (R', m, \varphi)) \quad (6.29)$$

From (6.28) and (6.29) we have that with non-negligible probability,

$$y_3 = y'_3 \text{ or } y_4 = y'_4 \quad (6.30)$$

Recall that (6.30) is the trigger condition for A to output 0. We conclude that when the VRF challenger's bit $b = 0$, the trigger condition for A to output 0 is met with non-negligible probability; and by construction, A outputs a random bit the rest of the time (i.e., when the trigger condition is not met).

**Case $b = 1$ in Parallel VRF Game.** In this case, the response that A receives to the challenge message $(R, m, \varphi)$ consists of truly random strings instead of VRF outputs, and so $\Pr[y'_3 = y_3 \lor y'_4 = y_4]$ is negligible. Thus, A outputs a random bit
with overwhelming probability.

We have shown that A's output is non-negligibly different depending on the VRF challenger's bit, and so A must win with non-negligible probability. This contradicts the security of the VRF. Therefore, R-RS is unforgeable.

Lemma 6.4.14 (Anonymity of R-RS). R-RS satisfies adaptive anonymity against adversarially chosen keys (Definition 6.3.4).

Proof (sketch). The proof is a hybrid argument using the security of VRFs and ZAPs to change the values $y_2$ and $y_4$ within the signature first to truly random values, then to VRF outputs w.r.t. party $i^*_0$ rather than $i^*_1$. Then the same procedure is applied to $y_1$ and $y_3$, so that finally $y_1, \ldots, y_4$ are all VRF outputs w.r.t. party $i^*_1$.

Suppose that this is not the case. Then there exists some $N = \text{poly}(k)$ and PPT $A = (A_1, A_2)$ such that the following probability is non-negligibly greater than $1/2$, where $I$ is the set of queries to the corruption oracle and $\mathcal{O} = \{\text{OSign}, \text{ORpd}\}$:

$$\Pr \left\{ \begin{array}{l}
    (vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \text{Gen}(1^k) \\
    ((m^*, i^*_0, i^*_1, R^*), s) \leftarrow A_1^{\mathcal{O}, \text{Corr}}(vk_1, \ldots, vk_N) \\
    b \leftarrow \{0, 1\} \quad : \quad b' = b \land \{i^*_0, i^*_1\} \cap I = \emptyset \\
    \sigma \leftarrow \text{Sign}(R^* \cup \{vk_{i^*_0}, vk_{i^*_1}\}, sk_{i^*}, m^*) \\
    b' \leftarrow A_2^{\mathcal{O}, \text{Corr}}(s, \sigma)
\end{array} \right\}.$$ 

We proceed via a sequence of hybrids.

**Hybrid 1.** The honest experiment, with $b = 0$.

**Hybrid 2.** Identical to the above, but in the generation of signature $\sigma = (\vec{\pi}, (y_1, \ldots, y_4))$, $y_2$ and $y_4$ are generated at random while $y_1$ and $y_3$ are still generated using the VRFs for party $i^*_0$.

By the security of the VRF, this hybrid is indistinguishable from Hybrid 1.

**Hybrid 3.** Identical to the above, but in the generation of signature $\sigma = (\vec{\pi}, (y_1, \ldots, y_4))$, $y_2$ and $y_4$ are generated using the VRFs for party $i^*_1$ rather than $i^*_0$. That is, parsing
\( sk_{i_1} = ((sk_{VRF}^{i_1}, \ldots, sk_{VRF}^{i_4}), v_k) \), we set

\[
y_2 = VRF.Eval(sk_{VRF}^{i_1}, (R, m, \varphi))
\]

and

\[
y_4 = VRF.Eval(sk_{VRF}^{i_4}, (R, m, \varphi)).
\]

By the security of the VRF, this hybrid is indistinguishable from Hybrid 2.

**Hybrid 4.** Identical to the above, but in the generation of signature \( \sigma \), the ZAPs in the Sign procedure are proven with respect to the witnesses for party \( i_1^* \) rather than \( i_0^* \).

That is, parsing \( sk_{i_1} = ((sk_{VRF}^{i_1}, \ldots, sk_{VRF}^{i_4}), v_k) \), in the invocation of Sign to generate \( \sigma \), let \( \tau_2' = VRF.Prove(sk_{VRF}^{i_1}, (R, m, \varphi)) \) and \( \tau_4' = VRF.Prove(sk_{VRF}^{i_4}, (R, m, \varphi)) \). For each \( i \in [N] \), in step 7 of Sign, generate \( \pi_i \) by invoking

\[
ZAP.Prove(\rho_1, stmt, (i_1^*, \bot, \tau_2', \bot, \tau_4', \gamma))
\]

instead of using the witness for \( i_0^* \).

By the witness indistinguishability property of the ZAP, this hybrid is indistinguishable from Hybrid 3.

**Hybrid 5.** Identical to the above, but in the generation of signature \( \sigma = (\bar{\pi}, (y_1, \ldots, y_4)) \), \( y_1 \) and \( y_3 \) are generated at random while \( y_2 \) and \( y_4 \) are still generated using the VRFs for party \( i_1^* \).

By the security of the VRF, this hybrid is indistinguishable from Hybrid 4.

**Hybrid 6.** Identical to the above, but in the generation of signature \( \sigma = (\bar{\pi}, (y_1, \ldots, y_4)) \), \( y_1 \) and \( y_3 \) (as well as \( y_2 \) and \( y_4 \)) are now generated using the VRFs for party \( i_1^* \).

By the security of the VRF, this hybrid is indistinguishable from Hybrid 5.

**Hybrid 7.** The honest experiment, with \( b = 1 \).

By the witness indistinguishability property of the ZAP, this hybrid is indistinguishable from Hybrid 6. \( \square \)
6.5 Claimable transformation

In this section, we give a black-box transformation from any ring signature to a claimable ring signature scheme. The transformation relies on one-way functions. If the original scheme was repudiable, the resulting scheme is moreover claimable-and-repudiable.

First, we define some building blocks.

6.5.1 Building blocks

We assume familiarity with the standard notions of standard (non-ring) signatures and PRFs, and simply establish syntax in this subsection.\(^\text{11}\)

**Definition 6.5.1 (Standard signature (syntax)).** A signature scheme is a triple of \(\Sigma = (\Sigma.\text{Gen}, \Sigma.\text{Sign}, \Sigma.\text{Verify})\) with the following syntax:

- \(\Sigma.\text{Gen}(1^k)\) takes as input the security parameter \(k\) and outputs a verification key \(vk\) and a signing key \(sk\).
- \(\Sigma.\text{Sign}(sk, m)\) takes as input a signing key \(sk\) and a message \(m\), and outputs a signature \(\sigma\).
- \(\Sigma.\text{Verify}(vk, \sigma, m)\) takes as input a verification key \(vk\), a signature \(\sigma\), and a message \(m\), and outputs a single bit indicating whether or not \(\sigma\) is a valid signature on \(m\) w.r.t. \(vk\).

**Definition 6.5.2. PRF (syntax)** A pseudorandom function (PRF) is a pair of algorithms \(\text{PRF} = (\text{PRF.Gen, PRF.Eval})\), where:

- \(\text{PRF.Gen}\) is a PPT algorithm that takes as input \(1^k\) and outputs a PRF key \(sk_{\text{PRF}}\), and
- \(\text{PRF.Eval}\) is a polynomial-time deterministic algorithm that takes as input a PRF key \(sk_{\text{PRF}}\) and \(x \in \{0, 1\}^*\) and outputs a string \(r\).

For simplicity, we assume PRFs that take arbitrary-length inputs (i.e., \(\{0, 1\}^*\)).

\(^\text{11}\)We refer to any standard textbook (e.g., [112]) for the relevant security definitions.
6.5.2 The transformation

Our transformation builds on any ring signature scheme, $\mathcal{RS}$, to construct a claimable ring signature scheme $\mathcal{C-RS}$. The basic idea is to take a signature $\sigma_{\mathcal{RS}}$ under $\mathcal{RS}$ and append to it a commitment $c$ to $(vk, \sigma_{\mathcal{RS}})$ where $vk$ is the verification key of the signer. The verification algorithm simply checks whether $\sigma_{\mathcal{RS}}$ verifies. The claim consists of a decommitment revealing that $c$ is a commitment to $(vk, \sigma_{\mathcal{RS}})$. Intuitively, by the hiding property of the commitment scheme, the identity of the signer is hidden until he chooses to publish a claim.

The simple transformation just described runs into a couple of problems when examined in detail. First, what if a signer commits to $(\sigma_{\mathcal{RS}}, vk')$ where $vk'$ is not his own key but that of someone else in the ring? This ability would violate equation (6.7) of Definition 6.3.8 (claimability). To prevent such behavior, our construction actually commits to a standard (non-ring) signature on $(vk, \sigma_{\mathcal{RS}})$. The unforgeability property of standard signatures then guarantees, intuitively, that a signer cannot convincingly make a claim with respect to any verification key unless he knows a corresponding signing key.

A second hurdle encountered by the scheme thus far described is that the signer must remember the commitment randomness in order to produce a claim. It is preferable that the signer need not be stateful in between signing and claiming; and indeed, recall that Definition 6.3.8 formalizes this property. To resolve this, our construction derives commitment randomness from a PRF. For similar reasons, the signing randomness for the standard (non-ring) signature in our construction is also derived from a PRF.

The formal description of the transformation follows.

**Construction 6.5.3.** Our transformation $\mathcal{C-RS}$ is parametrized by the following:

- $\mathcal{RS}$, a ring signature scheme,
- $\Sigma$, a standard signature scheme,
- $\text{Commit}$, a commitment scheme, and
- $\text{PRF}$, a PRF.
For convenience, and without loss of generality, we assume that the commitment randomness of $\text{Commit}$, the signing randomness of $\Sigma$, and the output of $\text{PRF.Eval}$ all have the same length of $\nu$ bits.

$\text{C-RS.Gen}(1^k)$

1. Let $(vk_{\text{RS}}, sk_{\text{RS}}) \leftarrow \text{RS.Gen}(1^k)$.
2. Let $(vk_{\Sigma}, sk_{\Sigma}) \leftarrow \Sigma.\text{Gen}(1^k)$.
3. Let $sk_{\text{PRF}} \leftarrow \text{PRF.Gen}(1^k)$.
4. Output $vk = (vk_{\text{RS}}, vk_{\Sigma})$ and $sk = (vk, sk_{\text{RS}}, sk_{\Sigma}, sk_{\text{PRF}})$. \(^{12}\)

In the rest of the construction, we implicitly parse verification keys and signing keys of $\text{C-RS}$ as $vk = (vk_{\text{RS}}, vk_{\Sigma})$ and $sk = (vk, sk_{\text{RS}}, sk_{\Sigma}, sk_{\text{PRF}})$ respectively. Also, for a ring

$$R = (vk_1 = (vk_{\text{RS}}^1, vk_{\Sigma}^1), \ldots, vk_N = (vk_{\text{RS}}^N, vk_{\Sigma}^N)),$$

we write $\text{RS}(R)$ to denote $(vk_{\text{RS}}^1, \ldots, vk_{\text{RS}}^N)$.

$\text{C-RS.Sign}(R, sk, m)$

1. Let $\sigma_{\text{RS}} \leftarrow \text{RS.Sign}(\text{RS}(R), sk_{\text{RS}}, m)$.
2. Let $r_{\Sigma} = \text{PRF.Eval}(sk_{\text{PRF}}, (vk, \sigma_{\text{RS}}, 0))$.
3. Let $\sigma_{\Sigma} = \Sigma.\text{Sign}(sk_{\Sigma}, (vk, \sigma_{\text{RS}}); r_{\Sigma})$.
4. Let $r_{\text{Commit}} = \text{PRF.Eval}(sk_{\text{PRF}}, (vk, \sigma_{\text{RS}}, 1))$.
5. Let $c = \text{Commit}((vk, \sigma_{\Sigma}); r_{\text{Commit}})$.
6. Output $\sigma = (\sigma_{\text{RS}}, c)$.

$\text{C-RS.Verify}(R, \sigma = (\sigma_{\text{RS}}, c), m)$

1. Output $\text{RS.Verify}(\sigma_{\text{RS}})$.

$\text{C-RS.Claim}(R, sk, \sigma = (\sigma_{\text{RS}}, c))$

1. Let $r'_{\Sigma} = \text{PRF.Eval}(sk_{\text{PRF}}, (vk, \sigma_{\text{RS}}, 0))$. \(^{12}\)See footnote 8.
2. Let \( r'_\text{commit} = \text{PRF.Eval}(sk_{\text{PRF}}, (vk, \sigma_{\text{RS}}, 1)) \).

3. Let \( \sigma'_\Sigma = \Sigma.\text{Sign}(sk_{\Sigma}, (vk, \sigma_{\text{RS}}); r'_\text{commit}) \).

4. If \( c \neq \text{Commit}(\sigma'_\Sigma, r'_\text{commit}) \), output \( \zeta = \bot \).

5. Otherwise, output \( \zeta = (r'_\text{commit}, \sigma'_\Sigma) \).

\[
\text{C-RS.VerClaim}(R, vk, \sigma = (\sigma_{\text{RS}}, c), \zeta = (r'_\text{commit}, \sigma'_\Sigma))
\]

1. Let \( c' = \text{Commit}((vk, \sigma'_\Sigma); r'_\text{commit}) \).

2. Output \( (c = c') \land \Sigma.\text{Verify}(vk_{\Sigma}, \sigma'_\Sigma, (vk, \sigma_{\text{RS}})) \).

**Theorem 6.5.4.** \( \text{C-RS} \) is a glorious ring signature scheme. Moreover, if \( \text{RS} \) is a repudiable ring signature scheme, then \( \text{C-RS} \) is repudiable-and-glorious.

**Lemma 6.5.5** (Correctness of \( \text{C-RS} \)). \( \text{C-RS} \) satisfies correctness (Definition 6.2.2).

Correctness is immediate, so we omit the proof.

**Lemma 6.5.6** (Claimability of \( \text{C-RS} \)). \( \text{C-RS} \) is claimable (Definition 6.3.8).

**Proof.** We show that \( \text{C-RS} \) satisfies each of the three conditions of Definition 6.3.8. The first condition is immediate by the correctness of the signature scheme \( \Sigma \), since the use of the \( \text{PRF} \) ensures that the values \( (r'_\Sigma, r'_\text{commit}, \sigma'_\Sigma) \) computed in \text{Claim} are the same as the corresponding values computed in \text{Sign} and that the commitments \( c, c' \) match.

For the second condition, assume for contradiction that there exists some PPT malicious claiming algorithm \( \text{MalClaim} = (\text{MalClaim}_1, \text{MalClaim}_2) \) that is able to claim a signature produced by a different party. That is, that the following probability is non-negligible:

\[
\begin{align*}
\Pr & \left[ (vk, sk) \leftarrow \text{C-RS.Gen}(1^k) \\
& (R', m) \leftarrow \text{MalClaim}_1^O(1^k, vk) \\
& \sigma \leftarrow \text{C-RS.Sign}(R' \cup \{vk\}, sk, m) \\
& (\zeta, vk') \leftarrow \text{MalClaim}_2^O(R' \cup \{vk\}, \sigma) \quad : \quad b = 1 \land b' = 1 \land vk' \neq vk \\
& b \leftarrow \text{C-RS.VerClaim}(R' \cup \{vk\}, vk', \sigma, \zeta) \\
& b' \leftarrow \text{C-RS.Verify}(R' \cup \{vk\}, \sigma, m) \end{align*}
\]

213
where $\mathcal{O} = \{\text{OSign}, \text{OClaim}_{\text{vk,sk}}\}$. We will produce an adversary $B$ that breaks the binding property of the commitment scheme. Let $B$ first run the experiment in Equation 6.31, invoking the malicious claiming algorithm $\text{MalClaim}$ and using its knowledge of the secret key $sk$ to answer the oracle queries of $\text{MalClaim}$. Let $B$ then sample $\zeta' \leftarrow \text{C-RS.Claim}(R' \cup \{vk\}, sk, \sigma)$ and $b'' \leftarrow \text{C-RS.VerClaim}(R' \cup \{vk\}, vk, \sigma, \zeta') = 1$. Since $\sigma$ is an honestly generated signature generated using key pair $(sk, vk)$, by the previous condition we have with all but negligible probability that $b'' = 1$. Consequently with non-negligible probability we have that $b = b'' = 1$ and $vk \neq vk'$. Conditioning on this event, we have that $\sigma = (\sigma_{\text{RS}}, c), \zeta = (r_{\text{commit}}, \sigma_{\Sigma}),$ and $\zeta' = (r'_{\text{commit}}, \sigma'_{\Sigma})$ are such that $c = \text{Commit}((vk, \sigma_{\Sigma}), r_{\text{commit}}) = \text{Commit}((vk', \sigma'_{\Sigma}), r'_{\text{commit}})$. But $vk \neq vk'$, so with non-negligible probability $B$ generates two different openings to the same commitment, breaking the binding property of the commitment. This concludes the proof of the second property of Definition 6.3.8.

For the third condition, assume for contradiction that there exists some PPT malicious signing-and-claiming algorithm $A$ that produces a signature and claims it on behalf of a different party. That is, assume that the following probability is non-negligible:

$$
\Pr\left[
\begin{aligned}
(vk, sk) &\leftarrow \text{C-RS.Gen}(1^k) \\
(R', m, \sigma, \zeta) &\leftarrow A^{O, \text{OClaim}_{\text{vk,sk}}(1^k, vk)}(1^k, vk) : b = 1 \land b' = 1 \\
b &\leftarrow \text{C-RS.VerClaim}(R' \cup \{vk\}, vk, \sigma, \zeta) \\
b' &\leftarrow \text{C-RS.Verify}(R' \cup \{vk\}, \sigma, m)
\end{aligned}
\right], \quad (6.32)
$$

where $\mathcal{O} = \{\text{OSign}\}$ and $Q$ is the set of queries made to the oracle $\text{OClaim}_{\text{vk,sk}}$. We will construct an adversary $B$ that breaks the unforgeability property of the signature scheme $\Sigma$. $B$ first requests a verification key $vk_{\Sigma}^*$ from the challenger for $\Sigma$. It samples keys $(vk, sk) \leftarrow \text{C-RS.Gen}(1^k)$ as in the beginning of the experiment in Equation 6.32, but replaces $vk_{\Sigma}$ with $vk_{\Sigma}^*$ and $sk_{\Sigma}$ with $\bot$ in $vk$ and $sk$, respectively. It then invokes the adversary $A$ on inputs $(1^k, vk)$ to produce values $(R', m, \sigma, \zeta)$, using the challenger for the signature scheme $\Sigma$ to produce the signatures $\sigma_{\Sigma}$ and $\sigma'_{\Sigma}$ used in the oracle
queries to \texttt{OSign} and \texttt{OClaim} except that the value \perp is returned whenever \texttt{OClaim} is queried on a signature that was not produced in response to a query to oracle \texttt{OSign}. Finally, it parses \( \zeta = (r'_\text{commit}, \sigma'_\Sigma) \) and outputs \( \sigma'_\Sigma \) as its forgery.

From the hiding property of the commitment scheme and the pseudorandomness of the PRF, it follows that except with negligible probability, \( A \) could not have produced a commitment \( c \) that passes the test of Step 4 of the \texttt{Claim} algorithm for the challenge identity \((vk, sk)\), unless the commitment \( c \) was the output of a previous invocation to oracle \texttt{OSign}. Consequently, except with negligible probability, the oracle responses will be distributed correctly, and so the value \( \zeta = (r'_\text{commit}, \sigma'_\Sigma) \) obtained is distributed identically to the value produced in the experiment of Equation 6.32. By assumption we have that with non-negligible probability, \( A \) never queried the oracle \texttt{OClaim}(vk, sk) on the signature \( \sigma = (\sigma_{RS}, c) \) and the \( \sigma'_\Sigma \) produced is such that \( \Sigma.\text{Verify}(vk_\Sigma, \sigma'_\Sigma, (vk, \sigma_{RS})) \) outputs 1, i.e. \( \sigma'_\Sigma \) verifies as a valid signature of the message \((vk, \sigma_{RS})\) under key \( vk_\Sigma \). It follows that with non-negligible probability, \( B \) successfully forges by producing a valid signature for a message on which the challenger was not queried. Consequently the third property of Definition 6.3.8 is also satisfied.

\begin{lemma} \textbf{(Unforgeability of C-RS).} C-RS is unforgeable (in the sense of Definition 6.3.10). \end{lemma}

\begin{proof}

The proof is by reduction to the unforgeability of RS.\footnote{Note that the unforgeability guarantee we have on \texttt{RS} is standard unforgeability of ring signatures (Definition 6.2.9), which does not give the adversary a \texttt{OClaim} oracle. If we had the stronger guarantee that \texttt{RS} were unforgeable in the presence of a \texttt{OClaim} oracle, then the unforgeability of C-RS would follow immediately, since signatures of C-RS contain signatures of \texttt{RS}.} Suppose, for contradiction, that there is a PPT adversary \( A \) that violates unforgeability of C-RS (Definition 6.3.10). Then we construct another adversary \( B \) that violates unforgeability of \texttt{RS} \textit{without having access to a \texttt{OClaim} oracle}. On input \((vk_1, \ldots, vk_N)\) which are verification keys of \texttt{RS}, \( B \) behaves as follows.

1. For each \( i \in [N] \):
   
   \begin{itemize}
   \item Sample \((vk_i^*, sk_i^*) \leftarrow \Sigma.\text{Gen}(1^\lambda)\).
   \end{itemize}

\end{proof}
• Sample \( sk_{PRF}^i \leftarrow PRF.Gen(1^\lambda) \).
• Let \( vk^*_i = (vk_i, vk_{\Sigma}^i) \) and \( sk^*_i = (sk_i, sk_{\Sigma}^i, sk_{PRF}^i) \).

2. Run A on input \((vk^*_1, \ldots , vk^*_N)\), answering A’s oracle queries as follows.

(a) For each query \((i, m, R)\) to C-RS.OSign:

• Query RS.OSign on \((i, m, R)\) and receive response \( \sigma_{RS} \).
• Let \( r_{\Sigma} = PRF.Eval(sk^i_{PRF},(vk^*_i, \sigma_{RS}, 0)) \).
• Let \( \sigma_{\Sigma} = \Sigma.Sign(sk^i_{\Sigma}, (vk^*_i, \sigma_{RS}); r_{\Sigma}) \).
• Let \( r_{Commit} = PRF.Eval(sk^i_{PRF}, (vk^*_i, \sigma_{RS}, 1)) \).
• Let \( c = Commit(\sigma_{\Sigma}; r_{Commit}) \).
• Output \( \sigma = (\sigma_{RS}, c) \).

(b) For each query \((i, R, \sigma)\) to C-RS.OClaim:

• Parse \( \sigma \) as \((\sigma_{RS}, c)\).
• Let \( r'_{\Sigma} = PRF.Eval(sk^i_{PRF}, (vk^*_i, \sigma_{RS}, 0)) \).
• Let \( r'_{Commit} = PRF.Eval(sk^i_{PRF}, (vk^*_i, \sigma_{RS}, 1)) \).
• Let \( \sigma'_{\Sigma} = \Sigma.Sign(sk^i_{\Sigma}, (vk^*_i, \sigma_{RS}); r'_{\Sigma}) \).
• Output \( \zeta = (r'_{Commit}, \sigma'_{\Sigma}) \).

3. Upon receiving an output \((R', m', \sigma')\) from A: parse \( \sigma' \) as \((\sigma'_{RS}, c')\), define \( R'' = \{vk_i : vk^*_i \in R'\} \), and output \((R'', m', \sigma'_{RS})\).

By construction of C-RS and B, whenever A successfully forges with respect to C-RS, B successfully forges with respect to RS. Moreover, B’s responses to A’s oracle queries are, by construction, distributed identically to the oracle responses in the unforgeability experiment for C-RS. Therefore, A’s probability of successful forgery is the same when B runs A in the above reduction, as in the unforgeability experiment. By our supposition, A’s forging probability in the unforgeability experiment is some non-negligible \( \epsilon \), so it follows that A’s forging probability in the above reduction is also \( \epsilon \), and therefore B’s forging probability with respect to RS is in turn \( \epsilon \). This contradicts the unforgeability of RS. The lemma follows. \( \square \)

**Lemma 6.5.8** (Anonymity of C-RS). If RS satisfies anonymity (resp., adaptive anonymity)
against adversarially chosen keys (Definition 6.2.5), then C-RS satisfies anonymity (resp., adaptive anonymity) against adversarially chosen keys (Definition 6.3.9).

Proof. We give the proof that C-RS satisfies adaptive anonymity whenever RS satisfies adaptive anonymity. The non-adaptive version of the statement has a slightly simpler proof: the proof follows the same structure, but certain steps of the proof become unnecessary. In the rest of the proof, we write "anonymity" to mean "adaptive anonymity against adversarially chosen keys."

We begin with a hybrid argument. Recall the anonymity experiment (Definition 6.2.3) for anonymity against adversarially chosen keys (Definition 6.3.9), shown below as "Hybrid 0."

\[
\text{Anonymity experiment (Hybrid 0)}
\]

\[
(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \text{Gen}(1^k)
\]

\[
((m^*, i_*^*, i_1^*, R^*), s) \leftarrow A_1^{\text{Sign, OClaim, Corr}}(vk_1, \ldots, vk_N)
\]

\[
b \leftarrow \{0, 1\}
\]

\[
\sigma \leftarrow \text{Sign}(R^* \cup \{vk_{i_0^*}, vk_{i_1^*}\}, sk_{i_*^*}, m^*)
\]

\[
b' \leftarrow A_2^{\text{Sign, OClaim}^{(\sigma)}, \text{Corr}}(s, \sigma)
\]

We now define two signing algorithms \text{Sign}_1 and \text{Sign}_2 which are slight variants of C-RS.Sign. For \(i \in \{1, 2\}\), we define Hybrid \(i\) to be the same as Hybrid 0 except that the invocation of \text{Sign} in the fourth line of the experiment is replaced by an invocation of \text{Sign}_i. In the descriptions of \text{Sign}_1 and \text{Sign}_2 below, changes from the preceding hybrid are marked in blue, and steps which are entirely removed are "crossed out" and shown in red.
Sign$_1(R, sk, m)$

1. Let $\sigma_{RS} \leftarrow \text{RS.Sign}(RS(R), sk_{RS}, m)$.
2. Let $r_\Sigma = \text{PRF.Eval}(sk_{PRF}, (vk, \sigma_{RS}, 0))$.
3. Let $\sigma_\Sigma = \Sigma.Sign(sk_\Sigma, (vk, \sigma_{RS}); r_\Sigma)$.
4. Let $r_{\text{Commit}} \leftarrow \{0, 1\}^\nu$.
5. Let $c = \text{Commit}(\sigma_\Sigma; r_{\text{Commit}})$.
6. Output $\sigma = (\sigma_{RS}, c)$.

Sign$_2(R, sk, m)$

1. Let $\sigma_{RS} \leftarrow \text{RS.Sign}(RS(R), sk_{RS}, m)$.
2. Let $r_\Sigma = \text{PRF.Eval}(sk_{PRF}, (vk, \sigma_{RS}, 0))$.
3. Let $\sigma_\Sigma = \Sigma.Sign(sk_\Sigma, (vk, \sigma_{RS}); r_\Sigma)$.
4. Let $r_{\text{Commit}} \leftarrow \{0, 1\}^\nu$.
5. Let $c = \text{Commit}(0; r_{\text{Commit}})$.
6. Output $\sigma = (\sigma_{RS}, c)$.

**Hybrid 1 is indistinguishable from Hybrid 0.** This follows from PRF security as long as there are no other variables in A’s view that are correlated with the PRF output in Hybrid 0, namely, $r_{\text{Commit}} = \text{PRF.Eval}(sk_{PRF}, (vk, \sigma_{RS}, 1))$. Since PRF security guarantees that PRF outputs on different inputs are computationally indistinguishable from uniform and independent strings, it suffices to establish that nowhere else in the anonymity experiment is the PRF evaluated on the specific input $(vk, \sigma_{RS}, 1)$. The only PRF evaluations during the experiment are to compute the $r_\Sigma$ and $r_{\text{Commit}}$ values used by the $O\text{Sign}$ oracle when responding to oracle queries. The PRF inputs used to compute $r_\Sigma$ values are distinct, by construction, by from those used to compute $r_{\text{Commit}}$ values (since the former end in 0 and the latter end in 1). The PRF inputs used to compute $r_{\text{Commit}}$ values for $O\text{Sign}$ queries are also, with overwhelming probability, distinct from the challenge input $(vk, \sigma_{RS}, 1)$. Since each such PRF input is of the format $(vk', \sigma'_{RS}, 1)$ where $\sigma'_{RS}$ is an honestly generated signature under RS, this follows from the following two observations.

1. The anonymity of RS implies that multiple honestly generated signatures under the same key pair must be distinct with overwhelming probability. (Otherwise, the adversary could break anonymity by querying a signature under every key in the ring on many messages, and then checking the challenge signature for equality with any of the preceding signatures.)

2. The unforgeability of RS implies that honestly generated signatures under an honestly generated key pair $(vk, sk)$ are distinct from honestly generated signatures under any other — possibly adversarially generated$^{14}$ — key pair.

$^{14}$With knowledge of $vk$ but not $sk$. 

218
Hybrid 2 is indistinguishable from Hybrid 1. This follows from the hiding property of the commitment, since all that has changed from Hybrid 1 is the value committed to by \( c \).

The rest of the proof gives a reduction between Hybrid 2 and the anonymity of RS. Note that the anonymity guarantee we have on RS is anonymity of standard ring signatures (Definition 6.2.9), which does not give the adversary a OClaim oracle.

Suppose, for contradiction, that there is a PPT adversary \( A = (A_1, A_2) \) that violates anonymity of C-RS (Definition 6.3.9). Then we construct another adversary \( B = (B_1, B_2) \) that violates anonymity of RS \textit{without having access to a OClaim oracle.}

On input \((v_{k_1}, \ldots, v_{k_N})\) which are verification keys of RS, \( B_1 \) behaves as follows.

1. For each \( i \in [N] \), construct \( v_{k_i}^* = (v_{k_i}, v_{k_i}^1) \) and \( s_{k_i}^* = (s_{k_i}, s_{k_i}^1, s_{k_i}^2) \) exactly as described in Step 1 in the proof of Lemma 6.5.7.
2. Run \( A \) on input \((v_{k_i}^1, \ldots, v_{k_N}^1)\), answering \( A \)'s oracle queries exactly as described in Step 2 in the proof of Lemma 6.5.7.
3. Upon receiving an output \(((m', i'_0, i'_1, R'), s')\) from \( A \) define \( R'' = \{v_{k_i} : v_{k_i}^* \in R'\} \), let \( s'' \) be all of \( B \)'s internal state, and output \(((m', i'_0, i'_1, R''), s'')\).

Then, on input \((s'', \sigma_{RS})\), \( B_2 \) behaves as follows.

1. Let \( c = \text{Commit}(0; r) \) for truly random \( r \).
2. Let \( \sigma = (\sigma_{RS}, c) \).
3. Run \( A_2 \) to obtain \( b' = A_2(s', \sigma) \).
4. Output \( b' \).

\( A \)'s view between the reduction run by \( B \) is identically distributed to \( A \)'s view in the experiment of Hybrid 2. Moreover, by construction, \( B_2 \)'s guess is correct exactly when \( A_2 \) guesses \( b' \) correctly. Therefore, \( B \)'s success probability in the anonymity experiment of RS is negligibly close to \( A \)'s success probability in the anonymity experiment of C-RS. By supposition, the latter probability is non-negligibly greater than \( 1/2 \). It follows that \( B \) violates the anonymity of RS, which is contradiction. The lemma follows. \( \Box \)
Theorem 6.5.9 (Claimability of C-RS). C-RS is a claimable ring signature scheme.

Proof. Follows from Lemmata 6.5.5–6.5.8. \qed

Theorem 6.5.10 (Repudiability-and-claimability of C-RS). If RS is repudiable, then C-RS is a repudiable-and-claimable ring signature scheme (Definition 6.3.14).

Proof. \qed

6.6 Unclaimable construction

In this section we show how to construct unclaimable ring signatures from lattice assumptions. The scheme is exactly the SIS-based ring signature scheme of Brakerski and Kalai [44], augmented with an additional algorithm ExtractRandomness.

We first give a summary of necessary background on lattice trapdoors; we refer the reader to [83] for a full description of the relevant algorithms.

6.6.1 Lattice trapdoor sampling

Let $q \in \mathbb{N}$, $m \in \mathbb{N}$, and $\beta \in \mathbb{Z}$ be functions of security parameter $n$. The (inhomogeneous, average-case) short integer solution (SIS$_{q,m,\beta}$) assumption states that given $A \leftarrow \mathbb{Z}_q^{n \times m}$, $v \leftarrow \mathbb{Z}_q^n$, it is computationally hard to find $x \in \mathbb{Z}_q^m$ such that $Ax = v$ and $\|x\| \leq \beta$. For polynomial $m$, $\beta$ and prime $q \geq \beta \cdot \omega(\sqrt{n \log n})$, the SIS problem is known to be as hard as approximating worst-case lattice problems, in particular the Shortest Independent Vectors Problem (SIVP), to within a factor of $\beta \cdot \tilde{O}(\sqrt{n})$ [139, 83].

Let $D_{\Lambda,s,c}$ denote the discrete Gaussian distribution over $n$-dimensional lattice $\Lambda$, centered at $c \in \mathbb{R}^n$ and with parameter $s$. We note the existence of the following algorithms, described in [83]$^{15}$:

- There is an algorithm TrapdoorSamp that on input a security parameter $1^n$ produces a matrix $A \in \mathbb{Z}_q^n$ and a trapdoor $T$, where $A$ is statistically close to uniform and $T$ is a short basis for the lattice $\Lambda^\perp(A)$.

$^{15}$These algorithms are given by algorithms TrapGen, SampleD and SampleISIS in [83].

220
There is an algorithm \textbf{SampleDist} that samples from the discrete Gaussian distribution $D_{ zm, s, 0}$.

There is an algorithm \textbf{SampleCond} that on input a matrix $A$, trapdoor $T$, parameter $s$ and vector $u$, produces a sample $x$ distributed statistically close to the discrete Gaussian distribution $D_{ zm, s, 0}$ conditioned on $Ax = u$. We have that $\|x\|_2 \leq s \sqrt{n}$ with probability 1.

We will also require additional algorithms that given output values of the algorithms \textbf{SampleDist} and \textbf{SampleCond}, respectively, sample randomness under which the algorithm produces the desired output.

- There is an algorithm \textbf{ExplainDist} that on input an image vector $x$ and parameter $s$, samples randomness $\rho$ that yields output $x$ under algorithm \textbf{SampleDist}, i.e. samples from the distribution $\{\rho | \text{SampleDist}(s; \rho) = x\}$.

- There is an algorithm \textbf{ExplainCond} that on input matrix $A$, trapdoor $T$, parameter $s$, vector $u$ and image vector $x$, samples randomness $\rho$ that yields output $x$ under algorithm \textbf{SampleCond} with inputs $(A, T, s, u)$, i.e. samples from the distribution $\{\rho | \text{SampleCond}(A, T, s, u; \rho) = x\}$.

We describe the algorithms \textbf{ExplainDist} and \textbf{ExplainCond} in Section 6.6.2. We additionally need the following lemma.

\textbf{Lemma 6.6.1.} Let $(A_1, T_1)$ and $(A_2, T_2)$ be sampled from \textbf{TrapdoorSamp}, and let $y \in \mathbb{Z}_q^n$. Sample vectors $x_1$ and $x'_2$ from \textbf{SampleDist}. Let $x_2 \leftarrow \text{SampleCond}(A_2, T_2, s, y - A_1x_1)$, and let $x'_1 \leftarrow \text{SampleCond}(A_1, T_1, s, y - A_2x'_2)$. Then the distributions $(A_1, T_1, A_2, T_2, x_1, x_2)$ and $(A_1, T_1, A_2, T_2, x'_1, x'_2)$ are statistically close.

Intuitively, this lemma says that the sampled vectors are distributed the same independently of which of the two trapdoors was used. This is closely related to the trapdoor extension technique of [49].
6.6.2 Explaining randomness of discrete Gaussian samples

Our construction requires "explaining" algorithms ExplainDist and ExplainCond. ExplainDist must, on input \((x, y)\), output \(\rho\) distributed according to the conditional distribution

\[
\left\{ \rho \mid \text{SampleDist}(x; \rho) = y \right\}.
\]

Similarly, ExplainCond must, on input \((x, y)\), output \(\rho\) distributed according to

\[
\left\{ \rho \mid \text{SampleCond}(x; \rho) = y \right\}.
\]

In this appendix, we outline how this "explaining randomness" is able to be efficiently computed according to the required distribution.

In our underlying instantiation, as in the original construction of [44]:

- SampleDist is instantiated by the function \(\text{SampleD}(B, s, c)\) defined in [82, §4.2].\(^{16}\)
- SampleCond is instantiated by the function \(\text{SampleISIS}(A, T, s, u)\) defined in [82, §5.3.2].

We now describe the relatively simple algorithms SampleD and SampleISIS in order to outline how one can efficiently compute randomness to explain any particular output. SampleISIS invokes SampleD, and SampleD in turn invokes a simpler algorithm SampleZ which is also defined in [82]. We describe these three algorithms in the order they are listed in the preceding sentence.

**Definition 6.6.2.** SampleISIS takes as input \((A, T, s, u)\) where \(A\) and \(T\) are matrices, \(s\) is a Gaussian parameter, and \(u\) is a vector.

1. By Gaussian elimination, choose an arbitrary \(t\) such that \(At = u \pmod{q}\).
2. Sample \(v \leftarrow \text{SampleD}(T, s, -t)\).
3. Output \(e = t + v\).

**Claim 6.6.3.** There is an efficient algorithm for ExplainCond if there is an efficient

\(^{16}\)We cite the full version here for the detailed description of the function; the conference version is [83].
algorithm for ExplainDist.

Proof. Essentially, the claim follows from the fact that Step 2 is the only randomized step in SampleISIS. Recall that ExplainCond needs to correctly sample the randomness distribution of SampleCond (i.e., SampleISIS) conditioned on a particular input and output. Given the input, t (Step 1) can be computed deterministically. Finally, given the output e and t, it remains only to explain the randomness of SampleD conditioned on input \((T, s, -t)\) and output \(e - t\). \(\square\)

**Definition 6.6.4.** SampleD takes as input \((B, s, c)\) where \(B = (b_1, \ldots, b_n)\) is a matrix describing a lattice basis, s is a Gaussian parameter, and c is a vector. In the following, tildes denote Gram-Schmidt orthogonalization.

1. Let \(v_0 := 0\) (the zero vector) and \(c_n := c\). For \(i = n, \ldots, 1\):
   (a) Let \(c'_i := \langle c_i, \tilde{b}_i \rangle / \langle \tilde{b}_i, \tilde{b}_i \rangle \in \mathbb{R} \) and \(s'_i = s / \|\tilde{b}_i\|_2 > 0\).
   (b) Sample \(z_i \leftarrow \text{SampleZ}(s'_i, c'_i)\).
   (c) Let \(c_{i-1} := c_i - z_i b_i\) and let \(v_{i-1} := v_i + z_i b_i\).
2. Output \(v_0\).

**Claim 6.6.5.** There is an efficient algorithm for ExplainDist if there is an efficient algorithm to "explain SampleZ" — i.e., an efficient algorithm that, on input \((s, c, x)\), samples from the following distribution:

\[
\left\{ \rho \mid \text{SampleZ}(s, c; \rho) = x \right\}.
\] (6.33)

Proof (sketch). Each output of SampleDist on a given input induces a unique set of \(z_i\)s — in other words, as noted in [82], there is a bijective correspondence between the random choices of the \(z_i\)s and the output. After recovering these \(z_i\)s, it remains only to explain the randomness of SampleZ conditioned on inputs \((s'_i, c'_i)\) and outputs \(z_i\) for each \(i\). \(\square\)

**Definition 6.6.6.** SampleZ takes input \((s, c)\) where \(s\) is a Gaussian parameter and \(c \in \mathbb{R}\). In the following, \(t(n) \in \omega(\sqrt{\log(n)})\) is some fixed function, say, \(t(n) = \log(n)\);
\text{Ber}(p) \text{ denotes the Bernoulli distribution with probability } p \text{ of outputting 1; and } \rho_s(\cdot) \text{ is the Gaussian measure, defined as } \rho_s(x) = \exp(-\pi \|x\|_2^2/s^2).

1. Let \( X = \mathbb{Z} \cap [c - s \cdot t(n), c + s \cdot t(n)] \) and sample uniformly \( x \leftarrow X \).
2. Sample \( b \leftarrow \text{Ber}(\rho_s(x - c)) \).
3. If \( b = 1 \), output \( x \). Else, go back to step 1.

**Claim 6.6.7.** There is an efficient algorithm that, on input \((s, c, x)\), samples from the distribution described in (6.33).

**Proof (sketch).** Sample\( Z \) consists of rejection sampling based on a biased coinflip, where the bias depends on the present sample \( x \). The randomness \( \rho \) to explain a particular output of Sample\( Z \) can be thought to consist of a series of “rejected” samples \( x_1, \ldots, x_{\ell - 1} \) followed by the “accepted” sample \( x_\ell \) (which must be equal to \( x \)). The length \( \ell \) of this sequence follows a geometric distribution parametrized by the expected bias \( \beta \) over the entire domain \( X \). That is,

\[
\beta = \frac{1}{|X|} \sum_{x \in X} \rho_s(x - c) .
\]

Once a value of \( \ell \) is sampled from the appropriately parametrized geometric distribution, it remains only to sample the “failed attempts” \( x_1, \ldots, x_{\ell - 1} \). This can be done by the following procedure. For each \( i \in [\ell - 1] \):

1. Sample uniformly \( x' \leftarrow X \).
2. Sample \( b \leftarrow \text{Ber}(\rho_s(x' - c)) \).
3. If \( b = 0 \), then set \( x_i := x' \) and continue. Else, go back to Step 1.
6.6.3 The basic construction of [44]

We now describe the construction of [44]. Brakerski and Kalai first construct a base version of their scheme that satisfies a weaker security notion, and augment it to fully-secure ring signatures in a series of steps. We now describe this base scheme.

Let the message space be \( \{0, 1\}^\ell \), and let \( X = \{ x \in \mathbb{Z}_q^m : \|x\|_2 \leq s\sqrt{m} \} \) for some \( s = \omega(\sqrt{n \log n \log q}) \) be the set describing what constitutes a "short" vector.

The key generation algorithm samples a matrix with an SIS trapdoor, and an additional set of \( 2\ell \) matrices, two corresponding to each bit of the message. It additionally samples a target vector \( y \), and outputs the matrices and target vector as the verification key and the trapdoor as the signing key.

\[
\text{BK-RS.Gen}(1^k)
\]

1. Let \( (A, T) \leftarrow \text{TrapdoorSamp}(1^k) \).
2. For \( (i, b) \in [\ell] \times \{0, 1\} \), let \( A_{i,b} \leftarrow \mathbb{Z}_q^{n \times m} \).
3. Let \( y \leftarrow \mathbb{Z}_q^n \).
4. Output \( vk = (A, (A_{i,b})_{(j,b) \in [\ell] \times \{0, 1\}, y}) \) and \( sk = (vk, T) \).

The signing algorithm proceeds as follows. A target vector \( y \) is selected from the lexicographically first verification key. For each identity in the ring, short vectors are sampled for matrices corresponding to each bit of the message to be signed, as well as the additional matrix. Finally, the trapdoor in the signing key is used to obtain a short vector, which is sampled from the same distribution conditioned on having a particular product with the matrix \( A_i \) corresponding to the signer, i.e. conditioned on Equation 6.34 being satisfied. The signature consists of the list of short vectors for each identity and each index of the message.

\[
\text{BK-RS.Sign}(R, sk, m; \rho)
\]

1. Parse \( R = (vk_1, \ldots, vk_N) \) and \( sk = (vk, T) \).

---

\(^{17}\) The original presentation of [44] introduces an abstraction they call ring trapdoor functions, and instantiates this abstraction from bilinear group assumptions as well as the SIS assumption. We present their SIS-based construction more explicitly, without the additional layer of abstraction, in order to make the role of the SIS trapdoor more apparent.
2. For \( i \in [N] \), parse \( v_k_i = (A_i, (A_{j,b})_{(j,b) \in \{\ell \times \{0,1\}}, y_i) \).

3. Let \( y = y_i \), where \( i \in [N] \) is the index for which \( v_k_i \) is lexicographically first.

4. If \( v_k \not\in R \), output \( \bot \) and halt.

5. Define \( i^* \in [N] \) be such that \( v_k_{i^*} = v_k \).

6. Using the trapdoor \( T_{A_i} \) for \( A_{i^*} \), we can sample \((x^{(i)}_j)_{i \in [N], j \in \{0\} \cup \ell}\) such that

\[
\sum_{i \in [N]} A_i x^{(i)}_0 + \sum_{i \in [N]} A^{(i)}_j x^{(i)}_j = y. \tag{6.34}
\]

That is, for \((i, j) \in [N] \times \{0\} \cup \ell\) other than the pair \((i^*, 0)\), we invoke algorithm \texttt{SampleDist} to sample \( x^{(i)}_j \in \) independently from the discrete Gaussian distribution \( X \). Finally, we invoke algorithm \texttt{SampleCond} use the trapdoor \( T \) for \( A_{i^*} \) to sample \( x^{(i^*)}_0 \) from a distribution statistically close to the distribution \( X \) conditioned on Equation 6.34 being satisfied.

7. Output \( \sigma = (x^{(i)}_j)_{i \in [N], j \in \{0\} \cup \ell} \).

The verification procedure simply checks that each vector in the signature has short entries and that Equation 6.34 is satisfied.

\texttt{BK-RS.Verify}(R, \sigma, m)

1. Parse \( R = (v_k_1, \ldots, v_k_N) \).
2. For \( i \in [N] \), parse \( v_k_i = (A_i, (A_{j,b})_{(j,b) \in \{\ell \times \{0,1\}}, y_i) \).
3. Parse \( \sigma = (x^{(i)}_j)_{i \in [N], j \in \{0\} \cup \ell} \).
4. For each \( x^{(i)}_j \) for \( i \in [N], j \in \{0\} \cup \ell \), if \( x^{(i)}_j \not\in X \) then immediately reject.
5. Let \( y = y_i \), where \( i \in [N] \) is the index for which \( A_{i^*} \) is lexicographically first.
6. Accept if Equation 6.34 above is satisfied, and otherwise reject.

Up to this point we have simply described the basic ring signature scheme of [44]. We now augment this scheme by providing an \texttt{ExtractRandomness} algorithm. In order to do so, we must produce “explaining randomness” that maps to the desired output.
vector under the algorithms $\text{SampleDist}$ and $\text{SampleCond}$. We do this by means of algorithms $\text{ExplainDist}$ and $\text{ExplainCond}$, as described in Appendix 6.6.2

$$\text{BK-RS.ExtractRandomness}(R, sk, \sigma, m)$$

1. Parse $R = (vk_1, \ldots, vk_N)$ and $sk = (vk, T)$.
2. For $i \in [N]$, parse $vk_i = (A_i, (A_{j,b})_{(j,b) \in \ell \times \{0,1\}}, y_i)$.
3. Parse $\sigma = (x_j^{(i)})_{i \in [N], j \in \ell \cup \{0\}}$.
4. If $vk \notin R$, output $\bot$ and halt.
5. Define $i^* \in [N]$ be such that $vk_{i^*} = vk$.
6. For $(i, j) \in [N] \times \{0\} \cup [\ell]$ other than the pair $(i^*, 0)$, invoke algorithm $\text{ExplainDist}$ to sample random coins $\rho_j^{(i)}$ that produce output $x_j^{(i)}$ under the discrete Gaussian sampling algorithm.
7. Invoke algorithm $\text{ExplainCond}$ to sample random coins $\rho_0^{(i^*)}$ that produce output $x_0^{(i^*)}$ under the conditional random sampling algorithm using trapdoor $T$.
8. Output $(\rho_j^{(i)})$.

**Theorem 6.6.8.** Assuming the $\text{SIS}_{q,m,\beta}$ assumption, BK-RS is a unclaimable ring signature scheme satisfying a weak notion of unforgeability in which the challenge is sampled at random at the beginning of the experiment.

**Sketch.** Completeness, anonymity and unforgeability are proven in [44]. It remains to show that the scheme is unclaimable. Consider the experiment described in Definition 6.3.12. For all indices $t$ other than the two $i$ and $i'$, the components of the signature corresponding to index $t$ are sampled from the same distribution, as are the components of the signature corresponding to matrices with no trapdoor. By the correctness of algorithm $\text{ExplainDist}$, the components of $\rho'$ and $\rho''$ corresponding to identities other than $i, i'$ are distributed statistically close, jointly with $R, S$, and the corresponding components of the signature. It remains to consider the portions of the signature corresponding to identities $i$ and $i'$. But Lemma 6.6.1 implies that the distribution of vectors $(x_0^{(i)}, x_0^{(i')})$ is statistically close, regardless of which trapdoor
was used to sample. By the correctness of algorithm \texttt{ExplainCond}, the corresponding components of \( \rho' \) and \( \rho'' \) are also statistically close, even conditioned on everything else. The conclusion follows.

6.6.4 Unclaimability for the full ring signature scheme of [44]

The ring signature construction we have just described satisfies a weak notion of unforgeability, in which the message on which a signature must be forged is sampled at random by the challenger and sent to the forger in the beginning of the experiment. In order to achieve full unforgeability, [44] proceed through a sequence of four reductions to construct schemes satisfying successively stronger notions of unforgeability. We now provide a brief overview of these reductions and describe how to modify the \texttt{ExtractRandomness} algorithm for each of these schemes.

The first modified scheme simply includes a description of the ring as part of the message to be signed. This only affects the message to be signed, and so the \texttt{ExtractRandomness} algorithm is unchanged and is simply invoked on a different message.

The second modification is the most complicated, and introduces a variant of chameleon hash functions. A chameleon hash function \( h \) is sampled during \texttt{Gen} and is included as part of the verification key \( \text{vk} \). During the \texttt{Sign} algorithm, randomness \( r \) is sampled from some distribution,\(^{18}\) and a value \( y = h(m, r) \) is computed, where \( m \) is the message to be signed and \( h \) is the hash function corresponding to the lexicographically first identity in the ring. The previous signature scheme is invoked on message \( y \), and the signature is augmented to include the randomness \( r \) as well. Observe that the only randomness to explain is the choice of \( r \) and the randomness used in the invocation of the previous signature scheme. Consequently the only modification to \texttt{ExtractRandomness} is that it now must also provide random coins resulting in a particular choice of the vector \( r \), which is straightforward.

The third modification simply computes a signature under the previous scheme

\[^{18}\text{The distribution over which } r \text{ is generated is uniform over the set of vectors with bounded } \ell_2 \text{ norm, i.e. } \{ r \in \mathbb{Z}_q^{m'} : 0 < \| r \|_2 \leq \beta \} \text{ for some } m', \beta \]
of every prefix of the message to be signed, and outputs a list of these $|m|$ signatures as its signature. We can invoke the previous ExtractRandomness algorithm for the previous scheme on each of these $|m|$ messages.

The final modification produces a random pad $\alpha$ as part of the Gen algorithm, and computes the signature on the exclusive or of the original message with the pad corresponding to the lexicographically first identity in the ring. This is the full ring signature scheme of [44]. As above, this only affects the message to be signed, and so the ExtractRandomness algorithm is simply invoked on a different message.

Given the ExtractRandomness algorithm for the weakly-unforgeable ring signature scheme in the previous section, the modifications we have just described yield a ExtractRandomness algorithm for the fully-unforgeable ring signature scheme of [44]. It is not difficult to see that this scheme satisfies Definition 6.3.11 and is an unclaimable ring signature scheme. Consequently, we obtain the following theorem.

Theorem 6.6.9. Assuming the SIS$_{q,m,\beta}$ assumption, the ring signature scheme of [44] combined with the ExtractRandomness algorithm described above is an unclaimable ring signature scheme.
Part IV

Incentivizing Collaboration
The next two chapters consider the use of cryptography — specifically, secure multi-party computation — to incentivize collaboration between mutually distrustful parties, towards societally beneficial outcomes from a game-theoretic perspective.

Of particular interest is the case where higher utility is achievable for all parties by collaborating, than by working alone. Even in such situations, collaboration may be practically hindered or prevented in the absence of cryptography, by the lack of trust between potential collaborators (and their unwillingness to rely on a “trusted third party” as a mediator).

Both of the papers presented in this part address this case by applying secure multiparty computation in conjunction with other cryptographic tools. Chapter 7 proposes and analyses a specific stylized model of data-based collaboration, designed to reflect some simplified properties of computations based on “big data” distributed across multiple parties in a competitive environment (e.g., for medical research, pharmaceutical development, or financial risk assessments). Chapter 8 considers the more abstract concept of equilibria in games, and proposes a way to achieve coarse correlated equilibrium — a generalization of the better-known correlated equilibrium, which can achieve higher utilities in certain classes of games — among distrusting parties using cryptography.

Data-Driven Collaboration Among Competing Parties

The availability of vast amounts of data is changing how we can make medical discoveries, predict global market trends, save energy, and develop new educational strategies. In certain settings such as Genome Wide Association Studies or deep learning, the sheer size of data (patient files or labeled examples) seems critical to making discoveries. When data is held distributedly by many parties, as often is the case, they must share it to reap its full benefits.

One obstacle to this revolution is the lack of willingness of different entities to share their data, due to reasons such as possible loss of privacy or competitive edge. Whereas cryptographic works address the data secrecy aspects, they shed no light on individual parties’ losses and gains when access to data carries tangible rewards.
Even if it is clear that better overall conclusions can be drawn from collaboration, are individual collaborators better off by collaborating? Addressing this question is the topic of this paper.

Our contributions are as follows.

- We formalize a model of $n$-party collaboration for computing functions over private inputs in which the participants receive their outputs in sequence, and the order depends on their private inputs. Each output “improves” on all previous outputs according to a score function.

- We say that a mechanism for collaboration achieves a collaborative equilibrium if it guarantees a higher reward for all participants when joining a collaboration compared to not joining it. We show that while in general computing a collaborative equilibrium is NP-complete, we can design polynomial-time algorithms for computing it for a range of natural model settings. When possible, we design mechanisms to compute a distribution of outputs and an ordering of output delivery, based on the $n$ participants’ private inputs, which achieves a collaborative equilibrium.

The collaboration mechanisms we develop are in the standard model, and thus require a central trusted party; however, we show that this assumption is not necessary under standard cryptographic assumptions. We show how the mechanisms can be implemented in a decentralized way by $n$ distrustful parties using new extensions of classical secure multiparty computation that impose order and timing constraints on the delivery of outputs to different players, in addition to guaranteeing secrecy and correctness.

**Coarse Correlated Equilibria Via Cryptography**

Nash equilibrium [144] and correlated equilibrium [8] are important solution concepts that have been extensively studied in both traditional and computational game-theoretic contexts. Coarse correlated equilibrium [143] is a closely related concept that was proposed as a generalization of correlated equilibrium, which can be more
powerful in some settings such as potential games.

The results of Chapter 8 apply methods from cryptography to enable any number of mutually distrusting players to implement the broad class of coarse correlated equilibria of strategic games without the need for trusted mediation.

Our implementation makes use of a (standard) pre-play “cheap talk” phase, in which players engage in free and non-binding communication prior to playing in the original game. In our cheap talk phase, the players execute a secure multi-party computation protocol to sample an action profile from an equilibrium of a “cryptographically blinded” version of the original game, in which actions are encrypted. The essence of our approach is to exploit the power of encryption to selectively restrict the information available to players about sampled action profiles, such that these desirable equilibria can be stably achieved. In contrast to previous applications of cryptography to game theory, this work is the first to employ the paradigm of using encryption to allow players to benefit from hiding information from themselves, rather than from others; and we stress that rational players would choose to hide the information from themselves.
Chapter 7

How to Incentivize Data-Driven Collaboration Among Competing Parties

This chapter presents results from “How to Incentivize Data-Driven Collaboration Among Competing Parties” [11], a joint work with Pablo Azar and Shafi Goldwasser that appeared in the proceedings of the 7th Innovations in Theoretical Computer Science conference (ITCS), 2016.

The availability of vast amounts of data is changing how we can make medical discoveries, predict global market trends, save energy, improve our infrastructures, and develop new educational strategies. Indeed, it is becoming clearer that *sample size* may be the most important factor in making surprising new discoveries in a number of areas such as *genome-wide association studies*\(^1\) (GWAS) and *machine learning* (ML), as witnessed by the striking success of GWAS studies with large samples for schizophrenia\(^2\) [36, 146, 77] and the success of deep learning in ML.

When large data is required, parts of the data are often held by different entities.

---

\(^1\)A genome-wide association study is an investigation of common genetic variants in a population, in order to identify genetic variants that are associated with a given trait.

\(^2\)“Dramatic increase in patient data size enabled the discovery of more than 100 gene loci associated with the disease up from a handful loci seen with small sets of patients. This was made possible due to an unusually large scale collaborations among many institutes.”
Such entities need to share their data, or at least engage in a collaborative computation where each entity manages its own private data, in order for society to reap the benefit of large sample sizes. Referring back to the GWAS example, success was explicitly attributed to such collaboration:

“The schizophrenia study was made possible due to unusually large scale collaborations among many institutes... This level of cooperation between institutions is absolutely essential... If we are to continue elucidating the biology of psychiatric disease through genomic research, we must continue to work together.” [104]

Unfortunately, the above example is the exception rather than the rule. A major obstacle to the big-data revolution is the lack of willingness of different entities to share data in collaborations with each other: so-called “data hoarding.” One obstacle is concerns about disclosure, where parties refuse to collaborate, in order to protect the secrecy of their data. Secrecy, however, is not the only obstacle.

An equally important obstacle is competition between entities holding data. When access to data carries tangible rewards, say, if the entities are companies competing for a share of the same market or research laboratories competing for scientific credit, it is unclear whether an individual collaborator is better off, even if it is clear that better overall conclusions can be drawn from collaboration. Stated in more game-theoretic terms, the entities face the following dilemma: whereas the overall societal benefit of collaboration is clear, the utility for an individual collaborator may be negative, so why collaborate? Addressing this question is the topic of this paper.

In this paper, we present a formal model for collaboration in which this question can be analyzed, as well as design mechanisms to enable collaboration where all collaborators are provably “better off,” when possible. The order in which collaborators receive the outputs of a collaboration will be a crucial aspect of our model and mechanisms. We believe that timing is an important and primarily unaddressed issue in data-based collaborations. For example, in the scientific research community, data sharing can translate to losing a prior publication date. In financial enterprises, the
timing of investments and stock trading can translate to large financial gains or losses.

We show in Section 7.3 that the collaboration mechanisms we develop can be implemented in a decentralized way by $n$ distrustful parties even in the presence of a subset of colluding polynomial-time parties who may deviate in an arbitrary fashion, under standard cryptographic assumptions. To achieve this, we extend the theory of multi-party computation (MPC) to impose order and time on the delivery of outputs to different players.

### 7.1 Overview of contributions

#### A model of collaboration

We propose a model for collaboration which enables the determination of whether the utility obtained by a collaborator outweighs the utility he may obtain without collaboration. The ultimate desired outcome of a collaboration is to learn a parameter of the (unknown) joint distribution from which the participants' input data $x_1, \ldots, x_n$ is drawn. This can be expressed as $y^* = f(\mathcal{X})$ where $\mathcal{X}$ is the joint distribution of input data and $f$ is a known function. In our model, the outcome of a collaboration is a pair $(\pi, \vec{Z})$ where $\pi$ is a permutation of player identities and $\vec{Z} = (Z_1, \ldots, Z_n)$ where each $Z_{\pi(i)}$ is a distribution that corresponds to player $i$'s “estimate” of $y^*$. We think of $Z_{\pi(i)}$ as the public output of player $i$: for example, in the setting of scientific collaboration, $Z_{\pi(i)}$ would be player $i$'s academic publication. Our model setup assumes an underlying score function which assigns scores to the players' outputs.

The model includes a reward function $R_t$ which characterizes the gain in utility for any given party $i$ in a collaboration. The reward that a party $i$ gets depends on how much his score $s(Z_{\pi(i)})$ improves on the previous state of the art $s(Z_{\pi(i-1)})$, and on $\pi(i)$, namely, when the party makes his public output. Specifically, the reward function includes a multiplicative discount factor $\beta^t$ where $\beta \in [0, 1]$ and $t$ is the time of publication, meaning that the reward from a publication is “discounted” more as time goes on.

$$R_t(\pi, \vec{Z}) = \beta^t \cdot (s(Z_{\pi(t-1)}) - s(Z_{\pi(t)}))$$
To determine whether the utility of collaboration outweighs the utility of working on one's own, our model uses "outside payoff" values $\alpha_i$ which are the score that party $i$ would obtain \textit{without collaborating}. $\alpha_i$ can be computed directly from the input $x_i$ of party $i$.

**Mechanisms and collaborative equilibrium**

We define a notion of \textit{collaborative equilibrium} in which all parties are guaranteed a non-negative reward, and develop \textit{mechanisms} for collaboration that compute such equilibria. When an equilibrium exists, our mechanism delivers a sequence of progressively improving "partial information" about $y^*$ to the collaborating parties. More specifically, the mechanism will take as input the data of all parties, and output a pair $(\pi, \hat{y})$ where $\pi$ is a permutation of player identities and $\hat{y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n)$ specifies the outcomes to be delivered to the players: each $\mathcal{Y}_{\pi(i)}$ is the approximation to $y^*$ that is given to player $i$ at time-step $\pi(i)$, such that the score of the outputs is increasing with time. That is, $s(\mathcal{Y}_{\pi(1)}) > \cdots > s(\mathcal{Y}_{\pi(n)})$. We emphasize that both the order $\pi$ and the outputs $\mathcal{Y}_i$ are computed based on the inputs of all players.

When player $i$ receives an output $\mathcal{Y}_{\pi(i)}$ from the central mechanism, she may combine $\mathcal{Y}_{\pi(i)}$ with the information that she learned from prior public outputs and her own input $x_i$, to generate a public output $Z_{\pi(i)}$. We first prove that the ability of the players to learn from others' publications, in general, will make the problem of deciding whether there exists an equilibrium is NP-complete (see Theorem 7.2.17).

Next, we show that there is a polynomial-time mechanism that can output an equilibrium whenever one exists (or output \textsc{NONE} if one does not exist) for a variety of model settings and parameters which we characterize (see Theorem 7.2.15). An example of a setting when a polynomial-time mechanism is possible is when

- there is an upper bound $\mu_j$ on the amount of information that any player can learn from a given player $j$'s publication, and
- it is possible to efficiently compute, for any $y^*$ and $\delta > 0$, an "approximation" $\mathcal{Y}'$ such that $s(\mathcal{Y}') = \delta$.  

238
In a nutshell, the bounds $\mu_j$ can be used to define a weighted graph in which the weight of the minimum-weight perfect matching determines the existence of a collaborative equilibrium.

**Cryptographic protocols to implement the mechanisms**

We develop cryptographic protocols for implementing the mechanisms without a centralized trusted party and in the presence of a subset of colluding players who may deviate from the protocol in an arbitrary fashion, under cryptographic assumptions. The protocols compute the collaboration outcome $(\pi, \mathcal{Y})$ via multi-party secure computation on players’ private inputs. Since a crucial aspect of the mechanism’s ability to yield non-negative reward to all players is the delivery of outputs in order, we need to extend the classical notion of MPC to incorporate guarantees on the order and timing of output delivery. These extensions may be of interest independent of the application of mechanisms for incentivizing collaborations.

We define *ordered MPC* as follows. Let $f$ be an arbitrary $n$-ary function and $p$ be an $n$-ary function that outputs permutation $[n] \to [n]$. An ordered MPC protocol is executed by $n$ parties, where each party $i \in [n]$ has a private input $x_i \in \{0, 1\}^*$, who wish to securely compute $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ where $y_i$ is the output of party $i$. Moreover, the parties are to receive their outputs in a particular *ordering* dictated by $p(x_1, \ldots, x_n) = \pi$ where $\pi$ is a permutation of the player identities. Since the choice of $\pi$ depends on private inputs, it may leak information: hence, we formulate an enhanced *secrecy* requirement for ordered MPC that each player should learn his output and his *own* position in the output ordering, and nothing more (see Definition 7.3.1).

We show a simple transformation from classical MPC protocols for general functionalities $f$ to ordered MPC protocols for general functionalities $f$ and permutation functions $p$ that achieve enhanced secrecy, even when a minority of the $n$ players may be colluding to sabotage the protocol (see Theorem 7.3.4). The assumptions necessary are the same as for the classical MPC constructions (e.g., [88]). When the colluding players are in majority, it is well known that output delivery to all honest
parties cannot be guaranteed [55].

Next, we define timed-delay MPC, where explicit time delays are introduced into the output delivery schedule. Time delays between the outputs may be crucial to enable parties to reap the benefits of their position in the order. We give two constructions of timed-delay MPC in the honest majority setting\(^3\). First, we give a conceptually simple protocol which runs “dummy rounds” of communication in between issuing outputs to different players, in order to measure time-delays. The simple protocol has the flaw that all (honest) players must continue to interact until the last party receives his output (that is, they must stay online until all the time-delays have elapsed). To address this issue, we present a second protocol assuming the existence of time-lock puzzles [155] in addition to the classical MPC [88] assumptions (see Theorem 7.4.6). Informally, a time-lock puzzle is a primitive which allows “locking” of data, such that it will be released after a pre-specified time delay, and no earlier. Our second timed-delay MPC protocol, instead of issuing outputs to players in the clear, gives to each party his output locked into a time-lock puzzle; and in order to enforce the desired ordering, the delays required to unlock the puzzles are set to be an increasing sequence. An issue that arises when giving out time-lock puzzles to many parties is that different parties may have different computing power, and hence solve their puzzles at different speeds: for example, it is clear that we cannot guarantee that players learn their outputs in the desired ordering if some players compute arbitrarily faster than others. Still, we show that our protocol is secure and achieves ordered output delivery in the case that the difference between any two players’ computing power is known to be bounded by a logarithmic factor. If the assumption about computing power does not hold, then the protocol still achieves security (i.e., correctness and secrecy), but the ordering of outputs is not guaranteed.

The definition of ordered and timed-delay MPC inspire new notions unrelated to the central topic of this paper. In particular:

- **Time-lines.** Inspired by the application of time-lock puzzles to time-delayed

\(^3\)We cannot hope to achieve timed-delay MPC in the case of dishonest majority since, as mentioned in the preceding paragraph, even output delivery cannot be guaranteed in this setting.
MPC, we propose the new concept of a *time-line*, where multiple data items can be locked so that their unlocking must be serialized in (future) time. See Section 7.5 for details.

- **Prefix-fairness.** In the traditional MPC landscape, fairness is the one notion that addresses the idea that either all parties participating in an MPC should benefit, or none should. Fairness requires that either all players receive their output, or none do. However, it is well-known that fairness is achievable when a majority of the players are honest, but it is *not* achievable for general functionalities when a majority of players are faulty [55]. We propose a refinement of the classical notion of fairness in the setting of ordered MPC, called *prefix-fairness*, where players are to receive their outputs one after the other according to a given ordering \(\pi\), and the guarantee is that *either* no players receive an output *or* those who do strictly belong to a prefix of the mandated order \(\pi\) (see Definition 7.3.3). Prefix-fairness can be achieved for general functionalities and *any number* of faulty players, under the same assumptions as classical MPC [88] (see Theorem 7.3.5).

### 7.1.1 Discussion and interpretation of our work

**Slowing down scientific discovery?** Intuitively, the mechanisms we develop always take the following form: the mechanism computes the “best possible estimate” \(Y^*\) of \(y^*\) given the input data of the players, and then hands out a sequence of successively more accurate (according to the score function) outcomes, where the final party receives \(Y^*\).

One may ask: why slow down scientific progress and hand out inferior results when better ones are available? We argue that progress will in fact be *enhanced*, not slowed down, by this methodology, as it will be a decisive factor in parties’ willingness to collaborate in the first place. This bears great similarity to the original philosophy of *differential privacy* and privacy-preserving data analysis more generally. In these fields, accuracy (so-called utility) of answers to aggregate queries over items in database is partially sacrificed in order to preserve privacy/secrecy of individual
data items, as a way to encourage individuals to contribute their data items to the database. In an analogous way, in order to get results based on the large data sets held by potential collaborators, we sacrifice the speed of discovery of the “ultimate” collaboration outcome: we are willing to pay this price to incentivize parties to collaborate and contribute their data. In contrast to differential privacy, we do not sacrifice ultimate accuracy. The last collaborator to receive an output, receives the ideal outcome $Y^\ast$. Namely, $Y_n = Y^\ast$.

**The Fort Lauderdale example: the importance of time.** A recurring idea in this work is the importance of time and ordering of research discoveries, which is inspired in part by the following striking example from the field of genomics. In the 2003 Fort Lauderdale meeting on large-scale biological research [185], the gathering of leading researchers in the field recognized that “pre-publication data release can promote the best interests of [the field of genomics]” but “might conflict with a fundamental scientific incentive — publishing the first analysis of one’s own data.” Researchers at the meeting agreed to adopt a set of principles by which although data is shared upon discovery, researchers hold off publication until the original holder of the data has published a first analysis. Being a close-knit community in which reputation is key, this was a viable agreement which has led to great productivity and advancement of the field. However, more generally, their report states that “incentives should be developed by the scientific community to support the voluntary release of [all sorts of] data prior to publication.” This example teaches us to focus on three key aspects of collaboration: the incentive to collaborate has to be clear to all collaborators; there must be a way to ensure adherence to the rules of collaboration; and timing is of the essence.

**Secrecy implies increased utility.** Although the goal of our work is to design mechanisms to incentivize collaboration by increasing the utility of collaborations rather than focusing on the secrecy of individual entities’ input data, MPC protocols prove to be an important technical tool to implement the mechanisms which guarantee
increased utility. As a by-product, the use of MPC provides our mechanisms with the additional guarantee of secrecy.

**Future directions.** When collaboration is feasible, each party $i$ in our model is guaranteed a reward from collaborating that is greater than the reward $a_i$ they could get on their own. However, the contributions of the players' data to the computation of the final output $Y^*$ may be asymmetric: some special player $i^*$ may have some data that helps solve the “puzzle,” but this player $i^*$ may not be known *a priori* before the participants decide to collaborate.\(^4\) An interesting future direction would be developing mechanisms where, even without *a priori* knowledge of which players have higher quality data, we can still design collaborations where the players whose contribution turned out most valuable get most credit.

Another future direction of interest to design *truthful* mechanisms so that collaborating parties will be provably incentivized to submit their true and accurate data as input. In our work, we assume that, while we can incentivize the players to collaborate or not, once they decide to collaborate they are truthful about the value of their dataset $x_i$. From the point of view of scientific publications, this assumption is reasonable if we believe that the experiments that generate this data can be verified or replicated, and that a failure to replicate would hurt a scientific group's reputation. However, there are many settings, such as businesses pooling their data together to generate larger profits, where the parties may be incentivized to lie about their output $x_i$. Since we are already assuming that parties are rational, a future direction would be to develop mechanisms where, even when parties can lie about $x_i$ (because $x_i$ cannot be verified by others), they are still incentivized to report it truthfully. One possible direction is where $x_i$ is the output of some long $\#P$ computation (for example, a Markov Chain Monte-Carlo simulation), where (a) replicating the computation would take a very long time and delay publication for everyone in the group and (b) player $i$ cannot prove in a classical way that their output $x_i$ is correct. Even

\(^4\)An example in the same vein is the following. In the medical setting, a hospital with a larger patient population will clearly have more patient data than a small facility, and yet access to data of small but homogeneous or rare communities can at times be more valuable than access to larger heterogeneous sets of data.
in this case, player $i$ can be incentivized to give the right answer via a rational proof [12, 13, 93].

Our setting is useful and most likely to lead to collaboration when there are increasing marginal returns from adding new data. It will be interesting to discover new settings where this is provably the case.

7.1.2 Other related work

The problem of how to make progress in a scientific community has been studied in other contexts. Banerjee, Goel and Krishnaswamy [16] consider the problem of partial progress sharing, where a scientific task is modeled as a directed acyclic graph of subtasks. Their goal is to minimize the time for all tasks to be completed by selfish agents who may not wish to share partial progress.

Kleinberg and Oren [115] study a model where researchers have different projects to choose from, and can work on at most one. Each researcher $i$ has a certain probability of being able to solve a problem $j$, and she gets a reward $w_j$ if she is the only person to solve it. If multiple researchers solve the problem, they study how to split the reward in a socially optimal way. They show that assigning credit asymmetrically can be socially optimal when researchers seek to maximize individual reward, and they suggest implementing a “Matthew Effect,” where researchers who are already credit-rich should be allocated more credit than in an even-split system. Interestingly, this is coherent with the results of our paper, where it is socially optimal to obfuscate data so that researchers who are already “ahead” (in terms of data), end up “ahead” in terms of credit.

Cai, Daskalakis and Papadimitriou [46] study the problem of incentivizing $n$ players to share data, in order to compute a statistical estimator. Their goal is to minimize the sum of rewards made to the players, as well as the statistical error of their estimator. In contrast, our goal is to give a decentralized mechanism through which players can pool their data, and distribute partial information to themselves in order so as to increase the utility of every collaborating player.

Boneh and Naor [41] construct timed commitments that can be “forced open” after
a certain time delay, and discuss applications of their timed commitments to achieve fair two-party contract signing (and coin-flipping) under certain timing assumptions including bounded network delay and the [155] assumption about sequentiality of modular exponentiation.

**Organization of the rest of the chapter.** Section 7.2 covers the scientific collaboration model, mechanisms, and feasibility theorems. Section 7.3 covers the definitions and constructions of ordered MPC, and Section 7.4 covers definitions and constructions of timed-delay MPC, and associated primitives such as time-line puzzles.

### 7.2 Data sharing model

In this section, we present and analyze mechanisms for scientific collaboration in our model. In our exposition, we focus primarily on the setting of scientific collaboration and publication. However, we want to highlight that our results apply to more broad collaboration and discovery in general, in which case a "publication" should be thought of as any kind of public output.

#### 7.2.1 The model

We propose a model of collaboration between $n$ research groups which captures the following features. Groups may pool their data, but each group will publish their own results. Moreover, only results that improve on the "state of the art" may be published. That is, a new result must improve on prior publications. However, more credit may be given to earlier publications. Finally, a group will learn not only from pooling their data with other groups, but also from other groups’ publications.

To formalize the intuitions outlined above, we specify a model as follows.

- There is a set $[n]$ of players.
- Each player $i$ has a dataset $x_i$ which is sampled as follows.
  - For each $i \in [n]$, there is a set $X_i$ of possible datasets, which is common
knowledge. Let $X$ denote $X_1 \times \cdots \times X_n$.
- There is a distribution $\mathcal{X} \in \Delta(X)$ over $X$, from which the $x_i$ are sampled:
  $(x_1, \ldots, x_n) \leftarrow \mathcal{X}$.
- The distribution $\mathcal{X}$ is not known to any of the players, but comes from a commonly known distribution $\mathcal{D}$. That is, $\mathcal{X} \leftarrow \mathcal{D}$, for some $\mathcal{D} \in \Delta(\Delta(X))$.

• There is an output space $Y$, and a function $f : \Delta(X_1 \times \cdots \times X_n) \rightarrow Y$ such that $\hat{y} = f(\mathcal{X})$ is the value which the players wish to learn. That is, the players want to learn some property of the unknown distribution $\mathcal{X}$ from which their datasets were sampled. $Y$ and $f$ are common knowledge.
• $\mathcal{Y}_0$ denotes the distribution of $\hat{y}$ given $f$ and $\mathcal{D}$.
• There is a score function $s : \Delta(Y) \rightarrow \mathbb{R}_+$, which varies with $f$ and $\mathcal{D}$. The score function $s(\cdot)$ is maximized by the distribution $\hat{\mathcal{Y}}$ which puts probability 1 on the true value $\hat{y}$. The score function $s$ is common knowledge.
• We require a natural monotonicity property of the score function. Namely, let $\mathcal{Y}$ and $\mathcal{Z}$ be any distributions, and let $z$ be a value in the support of $\mathcal{Z}$. Then
  \[ s(\mathcal{Y}) \leq s(\mathcal{Y}|z \leftarrow \mathcal{Z}), \]
  where $z \leftarrow \mathcal{Z}$ denotes the event that $z$ is sampled from the distribution $\mathcal{Z}$.
• **Remark.** Let $\{\hat{y}|x_1, \ldots, x_n\}$ denote the distribution of $\hat{y}$ given certain datasets $(x_1, \ldots, x_n) \in X$. A consequence of the monotonicity condition is that given all of the datasets $x_1, \ldots, x_n$ of all players in the model, the best achievable score is $s(\{\hat{y}|x_1, \ldots, x_n\})$.

• A collaboration outcome is given by a permutation $\pi : [n] \rightarrow [n]$ and a vector of output distributions $(\mathcal{Z}_1, \ldots, \mathcal{Z}_n) \in (\Delta(Y))^n$ such that $s(\mathcal{Y}_0) < s(\mathcal{Z}_{\pi(1)}) < \cdots < s(\mathcal{Z}_{\pi(n)})$.

  The intuition behind this condition is that, at time $t$, player $\pi(t)$ will publish $\mathcal{Z}_{\pi(t)}$. Since only results that improve on the “state of the art” can be published, we must have that the score $s(\mathcal{Z}_{\pi(t)})$ increases with the time of publication $t$.
• For a collaboration outcome $\omega = (\pi, \bar{Z})$, the player who publishes at time $t$
obtains a reward

$$R_t(\pi, \tilde{Z}) = \beta^t \cdot (s(\mathcal{Z}_{\pi(t)}) - s(\mathcal{Z}_{\pi(t-1)}))$$

where $\beta \in (0, 1]$ is a discount factor which penalizes later publications.\(^5\)

- For each player $i$, we define $\alpha_i = s(\{\hat{y}|x_i\}) - s(\mathcal{Y}_0) \in \mathbb{R}_+$, where $\{\hat{y}|x_i\}$ is the distribution of $\hat{y}$ given that the $i$th dataset is $x_i$. This models the “outside payoff” that player $i$ could get if she does not collaborate and simply publishes on her own.

- Players may learn information not only from their own data, but also from the prior publications of others. A learning bound vector $\{\xi_{\pi,i}\}_{\pi \in ([n] \rightarrow [n]), i \in [n]}$ characterizes, for any publication order $\pi$, the maximum amount that each player $i$ can learn from prior publications. This notion is defined formally in Section 7.2.3.

- We define $\mathcal{C}K$ to be the collection of all common-knowledge parameters of the model:

$$\mathcal{C}K = (\mathcal{D}, f, s, \beta).$$

\(^s\)

### 7.2.2 Examples

To illustrate the range of settings to which our model applies, we describe several concrete model instantiations.

Recall that our goal is to build mechanisms to enable collaborations by sharing data, in settings where such collaboration would be beneficial to all parties. Intuitively, such settings occur when the result that can be obtained based on the union of all players’ datasets is “much better” than the results that can be obtained based on the individual datasets: in other words, the “size of the pie” to be split between the collaborating players is at least as large as the sum of the “slices” obtained by players working individually. This intuition is made rigorous in Lemma 7.2.13, where we discuss score functions which satisfy a superadditivity condition (Property 7.2.12).

\(^5\)This is motivated by market scoring rules [95], where experts are rewarded according to how much they improve existing predictions.
Toy Example I (Secret-sharing). We begin with a “toy example” based on secret-sharing. This artificial first example is a dramatic illustration that the size of reward from collaboration can be much larger than the sum of individual rewards without collaborating.

Consider a stylized secret-sharing model with a secret \( \hat{y} \) drawn uniformly at random from \( \{0, 1\}^n \). Each player's data consists of a share \( x_i \in \{0, 1\}^n \) such that \( \hat{y} = x_1 \oplus \ldots \oplus x_n \) be the secret the players are trying to reconstruct. The shares are correlated and drawn from a distribution \( \mathcal{X} \) as follows:

- For each \( i \in [n-1] \), \( x_i \) is uniformly random in \( \{0, 1\}^n \).
- The last share is chosen such that \( x_n = \hat{y} \oplus x_1 \oplus \ldots \oplus x_{n-1} \).

The players want to learn \( f(X) = \hat{y} \). The score from publishing a distribution \( \mathcal{Y} \) is \( s(\mathcal{Y}) = H(\hat{y}) - H(\hat{y}|\mathcal{Y}) \) where \( H(\hat{y}) = n \) is the entropy of the uniformly random string \( \hat{y} \) and \( H(\hat{y}|\mathcal{Y}) \) is the entropy of \( \hat{y} \) given the distribution \( \mathcal{Y} \).

Without collaborating, each player \( i \) only knows a uniformly random string \( x_i \). Thus, \( H(\hat{y}|x_i) = H(\hat{y}) = n \) and \( \alpha_i = H(\hat{y}|x_i) - H(\hat{y}) = 0 \) for each player \( i \). Consider the following collaboration mechanism:

- Each player contributes share \( x_i \) to the mechanism.
- The mechanism computes \( \hat{y} = x_1 \oplus \ldots \oplus x_n \).
- The mechanism reveals \( i \)th digit \( \hat{y}_i \) to each player \( i \).

When participating in this mechanism, the first player will publish a guess \( \mathcal{Y}_1 \) which is a distribution over \( \{0, 1\}^n \) where the first bit of \( y \leftarrow \mathcal{Y}_1 \) is always \( \hat{y}_1 \). All other players learn \( \hat{y}_1 \) from player 1’s publication. Proceeding inductively, the \( i \)th player will publish a guess \( \mathcal{Y}_i \) such that the first \( i \) bits are correct, that is, \( (y_1, \ldots, y_i) = (\hat{y}_1, \ldots, \hat{y}_i) \) for any \( y \leftarrow \mathcal{Y}_i \). Note that since \( \alpha_i = 0 \) for each player \( i \), and \( H(\hat{y}|\mathcal{Y}_i) - H(\hat{y}|\mathcal{Y}_{i-1}) = 1 > \alpha_i \), this mechanism incentivizes players to collaborate.

Toy Example II (Network flow). Let \( G = (V, E) \) be a graph. Let \( \hat{s}, \hat{t} \in V \) be vertices which are connected by some number of disjoint paths. Consider a model where \( V, \hat{s}, \) and \( \hat{t} \) are common knowledge, and each player's data consists of a disjoint
subset of edges in $x_i \subseteq E$. More precisely, $(x_1, \ldots, x_n) \leftarrow \mathcal{X}(E)$ where $\mathcal{X}$ samples a partition of $E$.

The players want to learn the set of paths from $\tilde{s}$ to $\tilde{t}$. That is, $f(\mathcal{X}(E))$ is the set of paths in $E$ from $\tilde{s}$ to $\tilde{t}$. The score from publishing a distribution $Z$ over edges is

$$s(Z) = |\{p : p \text{ is a path in } E \text{ from } \tilde{s} \text{ to } \tilde{t}, \text{ and } \Pr_{z \sim Z} [p \subseteq z] = 1\}|.$$

In other words, the player’s score is given by how many paths from $\tilde{s}$ to $\tilde{t}$ she knows with certainty to exist in $E$. In some cases, it may be that no player knows any path from $\tilde{s}$ to $\tilde{t}$ based only on her own data, as illustrated by the simple example in the diagram below.

Consider the following collaboration mechanism:

- Each player contributes their edges $x_i$ to the mechanism.
- The mechanism computes $E = x_1 \cup \cdots \cup x_n$, and the set $P = \{p_1, \ldots, p_k\}$ of paths in $E$ that start at $\tilde{s}$ and end at $\tilde{t}$.
- The mechanism reveals the $i$th path $p_i$ to player $i$. If $k < n$, then the last $k - n$ players will get no output. If $k > n$, the “extra” paths are allocated arbitrarily to players.\(^6\)

When participating in this mechanism, the first player will publish a guess $Z_1$ which (always) samples the set $\{p_1\}$. All other players learn $p_1$ from player 1’s publication. Then, the $i$th player will publish a guess $Z_i$ that samples the set $\{p_1, \ldots, p_i\}$. As long as $s(Z_i) - s(Z_{i-1}) \geq \alpha_i$ for all $i \in [n]$ (note that this is the case in the diagram), this mechanism incentivizes players to collaborate.

\(^6\)It may be beneficial to allocate the “extra” paths strategically in order to reward players more fairly, or in order to make collaboration possible when the outside option values $\alpha_i$ are nonzero. However, in this example, we allocate them arbitrarily for simplicity.
Example III (Correlating gene loci with disease). This example is inspired by successful GWAS studies to identify gene loci associated with schizophrenia. Consider a model where each player holds a set of patients' medical (and in particular, genetic) data $x_i$ which comes from some unknown patient distribution $\mathcal{X}$. The players wish to learn the set $f(\mathcal{X})$ of gene loci that are correlated with the occurrence of schizophrenia in patients.

Let $\Gamma$ be the set of all gene loci. For $\gamma \in \Gamma$, define $1_{\gamma} = 1_{\gamma \in f(\mathcal{X})}$. The score from publishing a distribution $Z$ over $\mathfrak{P}(\Gamma)$ (i.e., over subsets of gene loci) could be:

$$s(Z) = \sum_{\gamma \in f(\mathcal{X})} \Pr[\gamma \in z] - \sum_{\gamma \notin f(\mathcal{X})} \Pr[\gamma \in z].$$

This score function rewards players for assigning high probabilities to gene loci $\gamma$ which are actually correlated with schizophrenia, and penalizes them for assigning high probabilities to those which are not. As in our previous examples, it turns out that in this setting, the reward that can be obtained based on pooling all the players' data is much greater than the sum of the rewards that could be obtained individually, as illustrated in Figure 7-1.

Consider the following collaboration mechanism:

- Each player contributes some patient data $x_i$.
- The mechanism computes $Y^* = \{f(\mathcal{X})|x_1, \ldots, x_n\}$, i.e., the distribution of $f(\mathcal{X})$ given all players' input data. Let $\Gamma^* = \{\gamma \in \Gamma: \Pr_{\mathfrak{P}(\Gamma)}[\gamma \in y] > 1/2\}$, that is, the set of gene loci that are more likely than not to be in $f(\mathcal{X})$, according to $Y^*$.
- The mechanism reveals to player $i$ the $i$th gene locus $\gamma_i$ in $\Gamma^*$. If $|\Gamma^*| < n$, then

---

7In practice, a more realistic scenario might be to model the extent to which particular gene loci are found to be correlated with the occurrence of schizophrenia, rather than classifying into binary categories “correlated” and “not correlated.” This case could be modeled, for example, by letting $f(\mathcal{X})$ be a vector $((\gamma_1, p_1), \ldots, (\gamma_N, p_N))$ where $\Gamma = \{\gamma_1, \ldots, \gamma_N\}$ is the set of gene loci, and for each $j \in [N]$, $p_j$ is the correlation coefficient between $\gamma_j$ and occurrence of schizophrenia. While Example IV presents the simpler “binary” model for ease of exposition, we remark that with appropriate modifications to the score function and mechanism, our model can accommodate the more complex case of estimating correlations, too.

8This is just one example of a reasonable mechanism for this model; we do not mean to claim that it is a canonical or optimal one. There are many variants which could make sense: for example, a simple modification would be to change the threshold $1/2$ in the second step.
Genome-wide association analysis identifies 13 new risk loci for schizophrenia.

- Genomewide association study identifies five new schizophrenia loci.
- Common polygenic variation contributes to risk of schizophrenia and bipolar disorder.

Figure 7-1: GWAS study success: the y-axis is the number of gene loci correlated with schizophrenia, and the x-axis is time (which corresponds to amount of data, since the reason for the improved findings was accumulation of data over time). Image ©Stephan Ripke

the last $k - n$ players will get no output. If $|\Gamma^*| > n$, the “extra” gene loci are allocated arbitrarily.\(^9\)

When participating in this mechanism, the first player will publish a guess $Z_1$ which (always) samples the set $\{\gamma_1\}$. All other players learn $\gamma_1$ from player 1’s publication. Then, the $i$th player will publish a guess $Z_i$ that samples the set $\{\gamma_1, \ldots, \gamma_i\}$.

Provided that $s(Z_i) - s(Z_{i-1}) \geq \alpha_i$ for all $i \in [n]$ (note that Figure 7-1 depicts exactly such a scenario), this mechanism incentivizes players to collaborate.

**Example IV (Statistical estimation).** Our last example is one where — in contrast to the examples so far — there are decreasing marginal returns from adding new information, and thus collaboration will not be feasible.

We consider a simple Bayesian model where the distribution $X$ is itself drawn from a “distribution over distributions” $D$. More concretely, each player $i$ receives a vector of $k_i$ samples $(x_{i,1}, \ldots, x_{i,k_i})$ drawn independently from a normal distribution $N(\mu, \sigma^2)$ with unknown mean $\mu$ and known variance $\sigma^2$. The mean $\mu$ is itself drawn

\(^9\)As remarked in Footnote 6, it can be beneficial to allocate the “extra” gene loci in a way which is not arbitrary, but instead optimized for making collaboration possible. In this example, for simplicity, we allocate them arbitrarily.
from a commonly known prior distribution $\mathcal{D} = N(m, 1)$ with known mean $m$ and variance 1. In this case, the ground set $X_i$ is $\mathbb{R}^{k_i}$. The distribution $\mathcal{X}(\mu, \sigma)$ is a product distribution over $\mathbb{R}^{\sum_{i=1}^{n} k_i}$, where each component of $(x_{i,1}, \ldots, x_{i,k_i})$ is drawn independently from $N(\mu, \sigma)$. The players want to learn $f(\mathcal{X}(\mu, \sigma)) = \mu$.

An estimator for $\mu$ is a random variable $\hat{\mu}$. The score of such a guess $\hat{\mu}$ is $s(\hat{\mu}) = -\mathbb{E}[(\hat{\mu} - \mu)^2]$. It is well known that if we have a vector $(x_{i,1}, \ldots, x_{i,k_i})$ of random samples drawn from $N(\mu, \sigma)$, the estimator that minimizes the expected squared error to $\mu$ is $\hat{\mu}_i = \frac{1}{k_i} \sum_{j=1}^{k_i} x_{i,j}$. Note that this is a normal random variable since each $x_{i,j}$ is sampled from normal random variable. The expectation of $\hat{\mu}_i$ is $\frac{1}{k_i} \cdot k_i \cdot \mu = \mu$ and the variance of $\hat{\mu}_i$ is $\frac{1}{k_i^2} \cdot k_i \cdot \sigma^2 = \frac{1}{k_i} \cdot \sigma^2$. Thus, $s(\hat{\mu}_i) = \frac{1}{k_i} \cdot \sigma^2$. If a player published by herself and did not collaborate, her reward would be the difference $\alpha_i = \sigma^2 - \frac{1}{k_i} \cdot \sigma^2$ between the priorly known variance $\sigma^2$ and the variance $\frac{1}{k_i} \cdot \sigma^2$ of player $i$'s estimate.

If the players collaborate, they can obtain the estimator $\hat{\mu}^* = \frac{1}{\sum_{i=1}^{n} k_i} \sum_{i=1}^{n} \sum_{j=1}^{k_i} x_{i,j}$ which has variance $s(\hat{\mu}^*) = \frac{1}{\sum_{i=1}^{n} k_i} \sigma^2$. The reward for $\hat{\mu}^*$ is the reduction in variance $\sigma^2 - s(\hat{\mu}^*) = \sigma^2 \cdot (1 - \frac{1}{\sum_{i=1}^{n} k_i})$. Note that in this case, the reward from an estimator only depends on the number of data points $N$ used to construct this estimator (in the above notation, $N = \sum_{i=1}^{n} k_i$). Furthermore, the reward $R(N) = \sigma^2 (1 - \frac{1}{N})$ that one could obtain with $N$ data points is concave in $N$. Intuitively, if one only has $N = 2$ data points, and gets 10 new ones, those 10 new data points are very valuable. However, if one already has $N = 2000000$ data points and gets 10 new ones, those 10 new data points do not increase the score very much.

This setting is in contrast to our Example III, where the score seemed to increase in a convex way with the number of data points. Indeed, in this Bayesian example, we will always have that

$$R(\sum_{i=1}^{n} k_i) = \sigma^2 (1 - \frac{1}{\sum_{i=1}^{n} k_i}) \leq \sigma^2 \sum_{i=1}^{n} (1 - \frac{1}{k_i}) = \sum_{i=1}^{n} R(k_i) .$$

In Section 2.5 we elaborate on why the above inequality is bad for collaboration. Intuitively, the left-hand side is the "size of the pie" if all players were to collaborate, and the right-hand side is the sum of the rewards that each player could receive.
on her own. The inequality implies there is no way to “slice the pie” so that every player has a bigger reward than the $\alpha_i$ they can get without collaborating, and thus collaboration is impossible.

In this simple Bayesian example, the marginal value of extra information will be decreasing. This raises the interesting question of when the value of information is (and is not) not convex with the amount of information available. For example, consider machine learning: learning problems whose objectives can be stated as minimizing a convex loss function (or maximizing a concave value function) seem to induce natural score functions which do not have increasing marginal returns, so our model may be more applicable to problems with non-convex objectives. We remark that such non-convex learning problems, in which our model seems more applicable, are an area of interest in machine learning as solving them is lately becoming practical—we refer to Bengio and LeCun [34] for a more thorough discussion of this situation.

7.2.3 Data-sharing mechanisms

We now return to the general formulation of our collaboration model, and we seek to design a general data-sharing mechanism that takes as input the data of all the parties, computes an output distribution $\mathcal{Y}_i \in \Delta(Y)$ for each $i \in [n]$, and outputs $\mathcal{Y}_i$ to each player $i$. The mechanism will output the $\mathcal{Y}_i$ values to players sequentially, in a particular order. Upon receiving $\mathcal{Y}_i$, player $i$ produces a public output (i.e a publication in the research collaboration example) which we denote by $Z_i \in Y$.

We note that the public output of player $i$ will not necessarily be the same as what was delivered by the data-sharing mechanism. Since player $i$ wants to maximize her reward, she will publish a result $Z_i$ that will maximize her reward, conditional on the information she has at the time of publication. This information includes, in addition to the output $\mathcal{Y}_i$ which she receives from the mechanism (and her knowledge of how the mechanism works\textsuperscript{10}), also her own dataset $x_i \in X_i$, and all the outputs $Z_j$ of other players that published before her.

Recall that a collaboration outcome $(\pi, \hat{Z})$ is given by a permutation $\pi : [n] \rightarrow \{1, \ldots, n\}$.

\textsuperscript{10}The mechanism description is common knowledge.
and a vector of output distributions \( \mathbf{Z} = (Z_1, \ldots, Z_n) \in (\Delta(Y))^n \) such that 
\[ s(Y_0) < s(Z_{\pi(1)}) < \cdots < s(Z_{\pi(n)}). \]
We now define a proposed collaboration outcome \( (\pi, \mathbf{Y}) \) as a permutation \( \pi : [n] \to [n] \) together with a vector of proposed outputs 
\( \mathbf{Y} = (Y_1, \ldots, Y_n) \in (\Delta(Y))^n \) generated by a data-sharing mechanism, satisfying 
\[ s(Y_0) < s(Y_{\pi(1)}) < \cdots < s(Y_{\pi(n)}). \]

Recall also that we need to bound how much player \( i \) can learn from previous publications (and from her own dataset). We formally capture this with the notion of learning bound vectors \( \xi_{\pi,i} \), which give an upper bound on the amount that player \( i \) learns from all previous publications when the order of publication is determined by permutation \( \pi \).

**Definition 7.2.1.** A learning bound vector \( \bar{\xi} = (\xi_{\pi,i})_{\pi \in ([n] \to [n]), i \in [n]} \) is a non-negative vector such that, if \( (\pi, \mathbf{Y}) \) is a collaboration outcome proposed by a data-sharing mechanism, and \( Z_i \) is the best (i.e., highest-scoring) distribution that player \( i \) can compute at the time \( \pi^{-1}(i) \) of her publication, then 
\[ s(Z_i) \leq s(Y_i) + \xi_{\pi,i}. \]
Let \( \Xi = \mathbb{R}^{n \times n}_+ \) denote the set of all learning bound vectors.

**Definition 7.2.2.** For a learning bound vector \( \bar{\xi} \), the set of inferred output distributions derived from a proposed collaboration outcome \( (\pi, \mathbf{Y}) \) is given by the following expression:
\[ \mathcal{I}_{\bar{\xi}}(\pi, \mathbf{Y}) = \{(Z_1, \ldots, Z_n) : \forall t \in [n], \ s(Y_{\pi(t)}) \leq s(Z_{\pi(t)}) \leq s(Y_{\pi(t)}) + \xi_{\pi,\pi(t)}\}. \]

The intuition behind the above definition is that the amount of information that player \( \pi(t) \) (namely, the player who publishes at time \( t \)) can learn from prior outputs is measured by how much her score increases based on these prior outputs. This increase in score is bounded by \( \xi_{\pi,\pi(t)} \). Thus, her eventual output will be some \( Z_{\pi(t)} \) with score between \( s(Y_{\pi(t)}) \) and \( s(Y_{\pi(t)}) + \xi_{\pi,\pi(t)} \).

**Remark 7.2.3.** In certain cases, \( \xi_{\pi,\pi(t)} \) measures exactly the amount of information that player \( \pi(t) \) can learn from her data. However, in our definition \( \xi_{\pi,\pi(t)} \) is an upper bound, and we emphasize that it may be a loose upper bound on the amount of
information $\pi(t)$ can learn. Our emphasis on this point comes from the following two reasons.

- In general, the vector $\vec{\xi} \in \mathbb{R}^{n \times n}$ has very high dimension, and finding such a vector is infeasible. We may want to approximate this vector via a low-dimensional encoding (as we will do below, where we encode learning bounds using $n$-dimensional vectors). Since this low-dimensional encoding will lose information, we will not be able to represent $\xi_{\pi, \pi(t)}$ exactly, but may get a reasonable upper bound on its value.

- For some other settings, we may not be able to derive a precise expression for $\xi_{\pi, \pi(t)}$ in terms of expectations, but we may still be able to derive an upper bound on the amount of information that player $\pi(t)$ learns.

Now that we have established a formal definition of learning bound vectors, we proceed to formally define a data-sharing mechanism.

Definition 7.2.4. For model parameters $\mathcal{E}$, a data sharing mechanism is a function

$$M : X \times \Xi \rightarrow ([n] \rightarrow [n]) \times (\Delta(Y))^n$$

which takes as inputs a vector $\vec{x} = (x_1, \ldots, x_n)$ of datasets and $\vec{\xi} = (\xi_{\pi, i})_{\pi \in ([n] \rightarrow [n]), i \in [n]}$ a learning bound vector, and outputs an ordering $\pi$ of the players and an output vector $(\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \in (\Delta(Y))^n$.

Remark 7.2.5. In the definition, for the sake of generality, we assume that the $\vec{\xi}$ values are given as input to the mechanism. We remark that in certain settings, these values can be computed directly from the inputs $x_i$ of the parties, as discussed in the examples of Section 1.1.2. In this case, one may think of the mechanism $M : X \rightarrow ([n] \rightarrow [n]) \times (\Delta(Y))^n$ as having input domain $X$ only.

7.2.4 Collaborative equilibria

In our model, each research group $\pi(t)$ will collaborate only if the credit they obtain from doing so is greater than the "outside option" reward $\alpha_{\pi(t)}$. We want to design a
mechanism that guarantees collaboration whenever possible. Accordingly, we define the following equilibrium concept.

**Definition 7.2.6.** Let \( CK \) be the model parameters. Let \((\vec{x}, \vec{\xi}) \in X \times \Xi\) and let \((\pi, (\mathcal{Y}_1, \ldots, \mathcal{Y}_n)) \in ([n] \rightarrow [n]) \times (\Delta(Y))^n\). We say that \((\pi, (\mathcal{Y}_1, \ldots, \mathcal{Y}_n))\) is a collaborative equilibrium with respect to \((\vec{x}, \vec{\xi})\) if for all inferred output distributions \(\vec{Z} = (Z_1, \ldots, Z_n) \in \mathcal{I}(\pi, (\mathcal{Y}_1, \ldots, \mathcal{Y}_n))\) and all \(t \in [n]\), it holds that \(R_t(\pi, \vec{Z}) \geq \alpha_{\pi(t)}\).

Our goal is to find data-sharing mechanisms for which collaboration is an equilibrium. Intuitively, since we are searching for a feasible permutation over a very high-dimensional space (\(n!\)-dimensional, to be precise), the problem will be NP-complete (this is proven in Theorem 7.2.17). However, there is a very natural condition on the learning vectors for which we can reduce the dimension of the search space and efficiently find a collaborative equilibrium. The feasible case corresponds to the case where, for any player \(j\), there is a bound on the amount of information that player \(j\) could teach any other players. We denote this bound by \(\mu_j\). Analogously, we could define \(\mu_j\) to be a bound on the amount that player \(j\) can learn from any other player. In this work, we describe only the first case, when \(\mu_j\) represents a bound on how much information player \(j\) can teach other players. The other case is analogous.

We define a learning bound vector to be \(n\)-dimensional if it satisfies the following property.

**Definition 7.2.7.** A learning vector \(\vec{\xi} \in \Xi\) is \(n\)-dimensional if there is a non-negative vector \((\mu_1, \ldots, \mu_n)\) such that \(\xi_{\pi(t)} = \sum_{\tau=1}^{t} \mu_{\pi(\tau)}\). Let \(\Xi_1 \subset \Xi\) denote the set of all \(n\)-dimensional learning vectors.

When \(\vec{\xi}\) is an \(n\)-dimensional learning vector, the total amount that player \(\pi(t)\) learns from all prior outputs is \(\sum_{\tau=1}^{t} \mu_{\pi(\tau)}\). In this case, we can give necessary and sufficient conditions for an equilibrium to exist (detailed in Theorem 7.2.9 below), provided that the following Output Divisibility Condition is satisfied.
Output Divisibility Condition. Given the model parameters \( \mathcal{X} \) and any real \( 0 < \delta \leq 1 \), there exists a distribution \( \mathcal{Y} \in \Delta(\mathcal{Y}) \) such that \( s(\mathcal{Y}) = \delta \).

**Remark 7.2.8.** The Output Divisibility Condition holds for a wide variety of natural score functions. In general, score functions which reward “how close” a distribution is to the true value \( \hat{y} = f(\mathcal{X}) \) will decrease (continuously) with the addition of random noise to a distribution. Provided that this holds, the Output Divisibility Condition can be satisfied by taking the optimal distribution \( \{\hat{y}|\mathcal{X}\} \) and perturbing it with random noise: the exact amount of noise to be added depends on the desired value of \( \delta \). To give a concrete example: in Example III (Gene loci), the perturbed distribution could simply add noise to the probabilities that each gene locus is sampled. Here, “adding noise” can mean simply adding some \( \eta \leftarrow N(0, \sigma^2) \) to the relevant parameters, where the magnitude of \( \sigma \) depends on the precise formulation of the score function and the desired value of \( \delta \).

**Theorem 7.2.9.** Suppose that the Output Divisibility Condition holds. Let \( \bar{x} \) be a vector of inputs and \( \bar{\xi} \) be an \( n \)-dimensional learning bound vector. Let \( \xi_{\pi,\pi(t)} = \sum_{t=1}^{t-1} \mu_{\pi(t)} \). Then for \((\pi, \bar{Y})\) to be a collaborative equilibrium, it is necessary and sufficient that

\[
\sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta^t} + \sum_{t=1}^{n} (n - t) \mu_{\pi(t)} \leq s(\mathcal{Y}_{\pi(n)}) - s(\mathcal{Y}_0).
\]

**Proof.** Necessary. Let \((\pi, \bar{Y})\) be a proposed collaborative equilibrium, and let \( \bar{Z} \in I(\pi, \bar{Y}) \) be a possible vector of inferred outputs. For every \( t \), we must have that:

\[
\beta^t \cdot (s(Z_{\pi(t)}) - s(Z_{\pi(t-1)})) \geq \alpha_{\pi(t)}.
\]

This is equivalent to:

\[
s(Z_{\pi(t)}) - s(Z_{\pi(t-1)}) \geq \frac{\alpha_{\pi(t)}}{\beta^t}.
\]

The worst case for player \( \pi(t) \) is when player \( \pi(t-1) \) learns as much as possible from

---

\(^{11}\)Recall (from the model description) that \( s(\{\hat{y}|\mathcal{X}\}) = \max_{\mathcal{Y} \in \Delta(\mathcal{Y})} s(\mathcal{Y}) \). Without loss of generality, we assume in our analysis that the score function is normalized so that its maximum value \( s(\{\hat{y}|\mathcal{X}\}) = 1 \).
prior publications and player π(t) learns as little as possible. That is, when

\[ s(\mathcal{Z}_\pi(t-1)) = s(\mathcal{Y}_\pi(t-1)) + \mu(1) + \cdots + \mu(\pi(t-2)) \quad \text{and} \quad s(\mathcal{Z}_\pi(t)) = s(\mathcal{Y}_\pi(t)). \]

In this case, the equilibrium condition becomes:

\[ s(\mathcal{Y}_\pi(t)) - s(\mathcal{Y}_\pi(t-1)) - \sum_{\tau=1}^{t-2} \mu(\tau) \geq \frac{\alpha(\pi(t))}{\beta^t}. \]

Rearranging slightly, we obtain:

\[ s(\mathcal{Y}_\pi(t-1)) - s(\mathcal{Y}_\pi(t)) \leq -\frac{\alpha(\pi(t))}{\beta^t} - \sum_{\tau=1}^{t-2} \mu(\tau). \]

Let us abuse notation slightly and define π(0) = 0. Then, summing over all t yields

\[ s(\mathcal{Y}_\pi(0)) - s(\mathcal{Y}_\pi(n)) \leq -\sum_{t=1}^{n} \frac{\alpha(\pi(t))}{\beta^t} - \sum_{t=1}^{n} (n - t)\mu(\tau). \]

Flipping the signs, the existence of a collaborative equilibrium implies:

\[ \sum_{t=1}^{n} \frac{\alpha(\pi(t))}{\beta^t} + \sum_{t=1}^{n} (n - t)\mu(\tau) \leq s(\mathcal{Y}_\pi(n)) - s(\mathcal{Y}_\pi(0)). \]

**SUFFICIENCY.** To prove that the condition is sufficient: given \( \mathcal{Y}_\pi(n) \) satisfying the inequality in the theorem statement, we need to construct \( \mathcal{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \) such that \((\pi, \mathcal{Y})\) is a collaborative equilibrium. We construct \( \mathcal{Y} \) inductively as follows: let \( \delta(\pi(n)) = s(\mathcal{Y}_\pi(n)) \), and for any t such that \( 2 \leq t \leq n \), let \( \delta(\pi(t-1)) = \delta(\pi(t)) - \frac{\alpha(\pi(t))}{\beta^t} - \sum_{\tau=1}^{t-2} \mu(\tau) \). Now that we have defined \( \{\delta(\pi(t))\}_{t=1}^{n} \) in this way, it follows that if we set \( \mathcal{Y}_\pi(t) \) such that \( s(\mathcal{Y}_\pi(t)) = \delta(\pi(t)) \), then for all \( t \geq 2 \) we have

\[ s(\mathcal{Y}_\pi(t)) - s(\mathcal{Y}_\pi(t-1)) = \delta(\pi(t)) - \delta(\pi(t-1)) = \frac{\alpha(\pi(t))}{\beta^t} + \sum_{\tau=1}^{t-2} \mu(\tau). \]

Note that it is possible to set \( \mathcal{Y}_\pi(t) \) in the required way, by the Output Divisibility Condition. Rearranging the above equation, it follows that:

\[ \beta^t \cdot (s(\mathcal{Y}_\pi(t)) - s(\mathcal{Y}_\pi(t-1)) - \sum_{\tau=1}^{t-2} \mu(\tau)) = \alpha(\pi(t)). \]
Since for any inferred outcome $Z_{\pi(t)}$ we have (by the definition of the learning bound vector) that

$$s(Y_{\pi(t)}) \leq s(Z_{\pi(t)}) \leq s(Y_{\pi(t)}) + \xi_{\pi,\pi(t-1)} = s(Y_{\pi(t)}) + \sum_{\tau=1}^{t-2} \mu_{\pi(\tau)},$$

we conclude that for all $t \geq 2$,

$$\beta^t \cdot (s(Z_{\pi(t)}) - s(Z_{\pi(t-1)})) \geq \alpha_{\pi(t)}.$$

Finally, we need to check that player $\pi(1)$ is incentivized to collaborate. Note that player $\pi(1)$ publishes first, so she cannot learn anything from previous publications. She will be incentivized to publish if

$$\beta \cdot (\delta_{\pi(1)} - s(Y_0)) \geq \alpha_{\pi(1)}.$$

This condition is equivalent to

$$\delta_{\pi(1)} - s(Y_0) \geq \frac{\alpha_{\pi(1)}}{\beta}.$$

Replacing $\delta_{\pi(t-1)} = \delta_{\pi(t)} - \frac{\alpha_{\pi(t)}}{\beta^t} - \sum_{\tau=1}^{t-2} \mu_{\pi(\tau)}$ iteratively, we get that player $\pi(1)$ is incentivized to collaborate if and only if

$$\delta_{\pi(n)} - s(Y_0) \geq \sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta^t} + \sum_{t=1}^{n} (n - t) \mu_{\pi(t)}$$

which is guaranteed by assumption. \qed

Recall from the definition of the score function that the best score that can be attained given datasets $x_1, \ldots, x_n$ is equal to $s(\{\tilde{y}|x_1, \ldots, x_n\})$. Based on Theorem 7.2.9, we can now characterize the datasets and learning bound vectors for which a collaborative equilibrium is possible.

**Definition 7.2.10.** Let $\mathcal{C} X$ be the model parameters and let $(\tilde{x}, \tilde{\xi}) \in X \times \Xi$. We say
that \((\bar{x}, \bar{\xi})\) supports a collaborative equilibrium if it holds that

\[
\sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta^t} + \sum_{t=1}^{n} (n - t) \mu_{\pi(t)} \leq s(\{\hat{y}|x_1, \ldots, x_n\}) - s(\mathcal{Y}_0) .
\]

**How do the model parameters affect feasibility of collaborative equilibria?**

Consider for a moment the simple case where \(\beta = 1\) and \(\bar{\xi} = \bar{\xi}_0\), that is, there is no discount factor and players do not learn from others’ publications. We can show that in this case, if the score function satisfies the following Property 7.2.12, then it holds that for all \(\bar{x} \in X\), \((\bar{x}, \bar{\xi})\) supports a collaborative equilibrium. That is, in this simple case, the condition for \((\bar{x}, \bar{\xi})\) to support an equilibrium reduces to the superadditivity of the auxiliary score function \(\bar{s}\) given in Property 7.2.12.

**Definition 7.2.11.** Let \(S\) be a set. A function \(f: S \to \mathbb{R}\) is superadditive if for all disjoint \(S_1, S_2 \subseteq S\), it holds that \(f(S_1) + f(S_2) \leq f(S_1 \cup S_2)\).

**Property 7.2.12** (Superadditive Differences). Let \(\mathcal{X}\) be the model parameters. We define an auxiliary score function \(\bar{s}: X_1 \cup \cdots \cup X_n \to \mathbb{R}_+\) which maps a set of datasets to a real-valued score, as follows:

\[
\bar{s}(\{(i_1, x_{i_1}), \ldots, (i_k, x_{i_k})\}) = s(\{\hat{y}|x_{i_1}, \ldots, x_{i_k}\}) - s(\mathcal{Y}_0) ,
\]

where \(\{\hat{y}|x_{i_1}, \ldots, x_{i_k}\}\) denotes the distribution of \(\hat{y}\) given that the datasets \(x_{i_1}, \ldots, x_{i_k}\) were sampled\(^{12}\) from \(\mathcal{X}\). The score function \(s\) satisfies the Superadditive Differences Property if \(\bar{s}\) is superadditive.

We observe that this precisely captures the intuition initially described in Section 7.2.2, that our model is designed to promote collaboration in situations where the reward that can be obtained from pooling all players’ data is more than the sum of the individual rewards that players can get.

\(^{12}\)More precisely: \(\{\hat{y}|x_{i_1}, \ldots, x_{i_k}\}\) is the distribution of \(\hat{y}\) given that each \(x_{i_j}\) was sampled in the \(i_j\)th position. (Recall that the distribution \(\mathcal{X}\) is over tuples of datasets \((x_1, \ldots, x_n)\).)
Lemma 7.2.13. Let $C_X$ be model parameters such that $\beta = 1$, let $\bar{x} \in X$ be arbitrary, and let $\xi = \bar{0} \in \Xi$. If $\bar{s}$ is a superadditive function on the input data, then $(\bar{x}, \xi)$ supports a collaborative equilibrium.

Proof. Recall the inequality from Definition 7.2.10:

$$\sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta^t} + \sum_{t=1}^{n} (n-t)\mu_{\pi(t)} \leq s(\{\tilde{y}|x_1, \ldots, x_n\}) - s(Y_0).$$

Since $\beta = 1$ and $\mu_{\pi(t)} = 0$, the left-hand side is simply $\sum_{t=1}^{n} \alpha_{\pi(t)}$. Using the definitions of $\alpha_{\pi(t)}$ and $\bar{s}$, and the fact that $\pi$ is a permutation, this can be rewritten as:

$$\sum_{t=1}^{n} \alpha_{\pi(t)} = \sum_{t \in [n]} (s(\{\tilde{y}|x_{\pi(t)}\}) - s(Y_0)) = \sum_{i \in [n]} \bar{s}(\{(i, x_i)\}).$$

Substituting back into the inequality, we obtain:

$$\sum_{i \in [n]} \bar{s}(\{(i, x_i)\}) \leq s(\{\tilde{y}|x_1, \ldots, x_n\}) - s(Y_0).$$

The right-hand side of the inequality is, by definition, equal to $\bar{s}(\{(1, x_1), \ldots, (n, x_n)\})$. Thus, the superadditivity of $\bar{s}$ implies that the inequality holds, and it follows that $(\bar{x}, \xi)$ supports a collaborative equilibrium.

Finally, we remark that either decreasing the discount factor $\beta$ or increasing the learning bound vector $\xi$ will make it harder to support a collaborative equilibrium (i.e., a lower value of $\beta$ means there will be fewer $(\bar{x}, \xi)$ which support an equilibrium), since these cause the left-hand side of the inequality to increase. So, while superadditivity is a sufficient condition in the simplest case, we observe that determining which $(\bar{x}, \xi)$ support a collaborative equilibrium is a more complex problem when the model parameters are varied.

### 7.2.5 The polynomial-time mechanism

We show a polynomial-time mechanism that computes a collaborative equilibrium in the case that learning bounds are given by a $n$-dimensional vector, provided that the
following *Efficient* Output Divisibility Condition is satisfied. The Efficient Output Divisibility Condition is a natural extension of the Output Divisibility Condition, which requires not only existence but also efficient computability of distributions with arbitrary score, while taking into account that the best possible score for given input datasets $x_1, \ldots, x_n$ is equal to $s(\{\hat{y}|x_1, \ldots, x_n\})$.

**Efficient Output Divisibility Condition.** Given model parameters $\mathcal{E}X$, datasets $x_1, \ldots, x_n \in X$, and any real $0 < \delta < s(\{\hat{y}|x_1, \ldots, x_n\})$, it is possible to efficiently compute a distribution $Y \in \Delta(Y)$ such that $s(Y) = \delta$.

**Remark 7.2.14.** The above condition holds for a wide variety of score functions, too: in particular, it holds for the class of score functions described in Remark 7.2.8. Suppose that the score function is continuous and decreases with the addition of random noise to a distribution. Then the condition can be satisfied by taking the "best computable" distribution $\{\hat{y}|x_1, \ldots, x_n\}$ and perturbing it with random noise: the amount of noise to add will depend on the desired value of $\delta$.

**Theorem 7.2.15.** Suppose the Efficient Output Divisibility Condition holds. Then there is a polynomial-time mechanism SHARE-DATA : $X \times \Xi_1$ that, given inputs $(\bar{x}, \bar{\mu})$ where $\bar{\mu} = (\mu_1, \ldots, \mu_n)$ represents a $n$-dimensional learning vector, outputs a collaborative equilibrium $(\pi, \bar{Y})$ whenever an equilibrium is supported by the inputs $(\bar{x}, \bar{\mu})$ (as defined in Definition 7.2.10), and outputs NONE otherwise.

**Proof.** The fact that the algorithm runs in polynomial time is immediate, since:

- additions, comparisons, and finding minimum weight matchings in a graph [70] can all be done in (randomized) polynomial time; and
- the Efficient Output Divisibility Condition implies that computing a distribution $\mathcal{Y}_{\pi(t-1)}$ such that $s(\mathcal{Y}_{\pi(t-1)}) = \delta_{\pi(t-1)}$ is efficient.

Recall that $(\bar{x}, \bar{\xi})$ supports a collaborative equilibrium if and only if there exists a
Algorithm 11 SHARE-DATA((x₁, ..., xₙ), (μ₁, ..., μₙ))

1. Let \( Y^* = \{ \hat{y}|x_1, ..., x_n \} \) and \( \delta^* = s(Y_0) \).
2. Construct a complete weighted bipartite graph \( G = (L, R, E) \) where \( L = [n], R = [n], E = L \times R \). For each edge \((i, t)\), assign a weight \( w(i, t) = \alpha \frac{\delta}{\beta} + (n - t)\mu_i \).
3. Let \( M \) be the minimum-weight perfect matching on \( G \). For each node \( t \in R \), let \( \pi(t) \in L \) be the node that it is matched with. If the weight of \( M \) is larger than \( \delta^* \), output NONE. Else, define \( \delta_{\pi(n)} = \delta^*, Y_{\pi(n)} = Y^* \).
4. For \( t \) from \( n \) to 2:
   - Let \( \delta_{\pi(t-1)} = \delta_{\pi(t)} - \frac{\alpha \delta}{\beta} - \sum_{t=1}^{t-2} \mu_{\pi(t)} \).
   - Let \( Y_{\pi(t-1)} \) be such that \( s(Y_{\pi(t-1)}) = \delta_{\pi(t-1)} \).
5. Output \( \omega = (\pi, (Y_{\pi(1)}, ..., Y_{\pi(n)})) \).

permutation \( \pi \) such that

\[
\sum_{t=1}^{n} \frac{\alpha \delta}{\beta^t} + \sum_{t=1}^{n} (n - t)\mu_{\pi(t)} \leq s(\{\hat{y}|x_1, ..., x_n\}) - s(Y_0).
\]

Note that our algorithm constructs a complete bipartite graph \( G = (L \cup R, E) \) where the weight on every edge is \( w(i, t) = \frac{\alpha \delta}{\beta} + (n - t)\mu_i \). A matching \( M \) on this graph induces a permutation \( \pi \) where, for every \( t \in R \), we have \( \pi(t) = i \) such that \((i, t) \in M \). The weight of such a matching is

\[
\sum_{t=1}^{n} \frac{\alpha \delta}{\beta^t} + \sum_{t=1}^{n} (n - t)\mu_{\pi(t)}.
\]

Thus, \((\tilde{x}, \tilde{\xi})\) supports a collaborative equilibrium if and only if the maximum-weight matching in \( G \) has weight less than or equal to \( w^* \triangleq s(\{\hat{y}|x_1, ..., x_n\}) - s(Y_0) \). Note that when the weight of the maximum-matching is greater than \( w^* \), our algorithm outputs NONE, indicating that an equilibrium is not supported by the inputs.

Finally, when the weight of the maximum matching is less than or equal to \( w^* \), the algorithm outputs a pair \((\pi, \tilde{Y})\) which (by construction) satisfies \( s(Y_{\pi(t-1)}) - s(Y_{\pi(t)}) = \frac{\alpha \delta}{\beta^t} + \sum_{t=1}^{t-2} \mu_{\pi(t)} \) for all \( t \in [n] \), so the sufficient conditions for \((\pi, \tilde{Y})\) to be a collaborative equilibrium are satisfied. \( \square \)
7.2.6 General NP-completeness

One may wonder if we can get an efficient mechanism for learning vectors which are not \( n \)-dimensional. We show that this is unlikely, since finding a collaborative equilibrium is NP-complete even under a weak generalization of \( n \)-dimensional learning vectors.

**Definition 7.2.16.** We say that a learning vector \( \xi \in \Xi \) is \( n^2 \)-dimensional if there exists a non-negative matrix \( (\mu_{i,j})_{(i,j)\in[n]\times[n]} \) such that \( \xi_{\pi(t)} = \sum_{\tau=1}^{t-1} \mu_{\pi(t),\pi(\tau)} \). We denote by \( \Xi_2 \subset \Xi \) the set of all \( n^2 \)-dimensional learning vectors.

When \( \xi \) is an \( n^2 \)-dimensional learning vector, the amount that player \( \pi(t) \) learns from \( \pi(\tau) \)'s output is bounded above by \( \mu_{\pi(t),\pi(\tau)} \). Thus, the total amount that player \( \pi(t) \) learns from all prior outputs is \( \sum_{\tau=1}^{t-1} \mu_{\pi(t),\pi(\tau)} \). The corresponding necessary condition for a collaborative equilibrium to be supported by some \((\vec{x},\vec{\xi})\) is that there is a permutation \( \pi \) such that

\[
\sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta^t} + \sum_{t=1}^{n} \sum_{s \neq t} \mu_{\pi(s),\pi(t)} \leq s(\{\hat{y}|x_1, \ldots, x_n\}) - s(Y_0).
\]

We show that even checking whether this condition holds is NP-complete.

**Theorem 7.2.17.** Given model parameters \( CK \), input datasets \((x_1, \ldots, x_n) \in X\), and a \( n^2 \)-dimensional learning bound vector \((\mu_{i,j})_{(i,j)\in[n]\times[n]}\), it is NP-complete to decide whether there exists \( \pi \) such that

\[
\sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta^t} + \sum_{t=1}^{n} \sum_{s > t} \mu_{\pi(s),\pi(t)} \leq s(\{\hat{y}|x_1, \ldots, x_n\}) - s(Y_0).
\]

**Proof.** It is clear that the problem is in NP, since given a permutation \( \pi \), the left-hand side can be efficiently computed and compared to the right-hand side of the inequality.

To show that the problem is NP-hard, we reduce it to the *minimum weighted feedback arc set problem*. The unweighted version of this problem was shown to be NP-complete by Karp [111], and the weighted version is also NP-complete [71].

264
All that we need to show is that, given a graph \( G \), a set \( S \) of edges is a feedback arc set if and only if there exists a permutation \( \pi \) of the vertices of \( V \) such that
\[
S = \{ (\pi(t), \pi(s)) \in E : s < t \}.
\]

To see this, note that if \( \pi \) is a permutation and
\[
S = \{ (\pi(t), \pi(s)) \in E : s < t \}
\]
then the set \( S \) intersects every cycle of \( G \). This is because if
\[
C = \{ (\pi(i_1), \pi(i_2)), (\pi(i_2), \pi(i_3)), \ldots, (\pi(i_k), \pi(i_1)) \}
\]
is a cycle in \( G \), then there must exist \( s, t \) such that \( s < t \) and \( (\pi(t), \pi(s)) \in C \), so \( S \) intersects \( C \). Thus, \( S \) is a feedback arc set.

Conversely, if \( S \) is a feedback arc set, then \( G' = (V, E - S) \) is a directed acyclic graph, and we can induce an ordering \( \pi \) on \( V \) following topological sort. Any edge \( (\pi(t), \pi(s)) \in E - S \) must satisfy \( t < s \). Thus, any edge \( (\pi(t), \pi(s)) \) where \( s < t \) must be in \( S \). Thus, given \( \pi \) from the topological sort, we must have
\[
S \supset \{ (\pi(t), \pi(s)) \in E : s < t \}.
\]
Since weights are non-negative, the minimal feedback arc set \( S^* \) will correspond to a permutation \( \pi^* \) such that
\[
S^* = \{ (\pi^*(t), \pi^*(s)) \in E : s < t \}.
\]

We show how to reduce Minimum-Weight-Feedback-Arc-Set to our problem. Given \( G = (V, E) \), \( w : E \to \mathbb{R}_{\geq 0} \) and \( t \in \mathbb{R}_{\geq 0} \), let
\[
s(\{\hat{y}|x_1, \ldots, x_n\}) - s(\mathcal{Y}_0) = \gamma \quad \text{and let} \quad \mu_{i,j} = w(i,j) \text{ if } (i,j) \in E \text{ and } \mu_{i,j} = 0 \text{ otherwise.}
\]

Suppose there exists a permutation \( \pi \) such that
\[
\sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta} + \sum_{t=1}^{n} \sum_{s > t} \mu_{\pi(s), \pi(t)} \leq s(\{\hat{y}|x_1, \ldots, x_n\}) - s(\mathcal{Y}_0).
\]
Then, since the $\alpha_i$ and $\beta$ are positive,

$$\sum_{t=1}^{n} \sum_{s>t} \mu_{\pi(s),\pi(t)} \leq s(\{y\vert x_1, \ldots, x_n\}) - s(\mathcal{Y}_0).$$

Plugging in our choices of $\mu_{i,j}$ and $s(\{y\vert x_1, \ldots, x_n\}) - s(\mathcal{Y}_0)$, this becomes

$$\sum_{(\pi(s),\pi(t)) \in E : s > t} w(\pi(s), \pi(t)) \leq \gamma.$$

Since the set $S = \{(\pi(s), \pi(t)) \in E : s > t\}$ is a feedback arc set, we have that there exists a feedback arc set with weight less than $\gamma$.

Conversely, assume no such permutation $\pi$ exists. That is,

$$\sum_{(\pi(s),\pi(t)) : s > t} \mu_{\pi(s),\pi(t)} > s(\{y\vert x_1, \ldots, x_n\}) - s(\mathcal{Y}_0)$$

for all permutations $\pi$. Note that whether $s$ comes before $t$ or vice-versa does not matter, since this inequality holds for all permutations. Thus, we can also write

$$\sum_{(\pi(s),\pi(t)) : s < t} \mu_{\pi(s),\pi(t)} > s(\{y\vert x_1, \ldots, x_n\}) - s(\mathcal{Y}_0)$$

for all permutations $\pi$. From the argument above, the minimum weight feedback arc set $S^*$ induces a permutation $\pi^*$ such that $S^* = \{(\pi^*(t), \pi^*(s)) \in E : s < t\}$. The weight of $S^*$ is

$$\sum_{(\pi^*(s),\pi^*(t)) \in E : s < t} w(\pi^*(s), \pi^*(t)) = \sum_{(\pi^*(s),\pi^*(t)) \in E : s < t} \mu_{\pi^*(s),\pi^*(t)} > s(\{y\vert x_1, \ldots, x_n\}) - s(\mathcal{Y}_0) = \gamma.$$

Thus, there does not exist a feedback arc set with weight less than or equal to $\gamma$.

We conclude that if we can efficiently check whether

$$\sum_{t=1}^{n} \frac{\alpha_{\pi(t)}}{\beta^t} + \sum_{t=1}^{n} \sum_{s>t} \mu_{\pi(s),\pi(t)} \leq s(\{y\vert x_1, \ldots, x_n\}) - s(\mathcal{Y}_0),$$

266
then we can efficiently check whether there exists a feedback arc set $S$ with weight less than $\gamma$. Thus, the feedback arc set problem reduces to ours, and our problem is NP-complete.

We have shown that in our model of scientific collaboration, it can indeed be very beneficial to all parties involved to collaborate under certain ordering functions, and such beneficial collaboration outcomes can be efficiently computed under certain realistic conditions (but probably not in the general case).

### 7.3 Ordered MPC

We introduce formal definitions of ordered MPC and associated notions of fairness and ordered output delivery, and give protocols that realize these notions. Our definitions build upon the standard security notion\textsuperscript{13} for traditional MPC (Section 2.7).

Throughout this work, we consider computationally bounded (rushing) adversaries in a synchronous complete network, and we assume the players are honest-but-curious, since any protocol secure in the presence of honest-but-curious players can be transformed into a protocol secure against malicious players [88].

#### 7.3.1 Definitions

Let $f$ be an arbitrary $n$-ary function and $p$ be an $n$-ary function that outputs permutation $[n] \to [n]$. An ordered MPC protocol is executed by $n$ parties, where each party $i \in [n]$ has a private input $x_i \in \{0, 1\}^*$, who wish to securely compute $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n) \in (\{0, 1\}^*)^n$ where $y_i$ is the output of party $i$. Moreover, the parties are to receive their outputs in a particular ordering dictated by $p(x_1, \ldots, x_n) = \pi \in ([n] \to [n])$. That is, for all $i < j$, party $\pi(i)$ must receive his output before party $\pi(j)$ receives her output. Note that the output ordering $\pi$ is \textit{data-dependent}, as $p$ is a function of the parties’ inputs.

\textsuperscript{13}Note that throughout this work, we use “stand-alone” security notions rather than “universally composable” ones.
Following [88], the security of ordered MPC with respect to a functionality $f$ and permutation function $p$ is defined by comparing the execution of a protocol to an ideal process $F_{\text{Ordered-MPC}}$ where the outputs and ordering are computed by a trusted party who sees all the inputs. An ordered MPC protocol $F$ is considered to be secure if for any real-world adversary $A$ attacking the real protocol $F$, there exists an ideal adversary $S$ in the ideal process whose outputs (views) are indistinguishable from those of $A$. Note that this implies that no player learns more information about the other players' inputs than can be learned from his own input and output, and his own position in the output delivery order. The latter condition is important because the output ordering depends on parties' private inputs, and thus we require that the protocol reveals as little information as possible about the ordering.

**Many rather than one view.** In the ordered MPC setting, the ideal adversary $S$ and the real-world adversary $A$ each output a view after each output phase. This is in contrast to standard MPC, where the adversaries simply output one view at the end of the protocol execution.

**Definition 7.3.1 (Security).** A multi-party protocol $F$ is said to securely realize $F_{\text{Ordered-MPC}}$, if the following conditions hold.

1. The protocol description specifies $n$ check-points $C_1, \ldots, C_n$ corresponding to events during the execution of the protocol.
2. Take any PPT adversary $A$ who corrupts a subset of players $S \subset [n]$, and let $V_{A,j}$ be the result of an arbitrary function $A$ applies to his view after each check-point $C_j$. Let

$$V_A^{\text{real}} = ((V_{A,1}, V_{1,1}, \ldots, V_{n,1}), \ldots, (V_{A,n}, V_{1,n}, \ldots, V_{n,n}))$$

be the tuple consisting of the adversary $A$'s outputted views along with the outputs of the real-world parties as specified in the ideal functionality description. Then there is a PPT ideal adversary $S$ which, attacking $F_{\text{Ordered-MPC}}$ by corrupting the same subset $S$ of players, can output views $V_{S,j}$ such that for each $j \in [n]$, it holds that $V_{S,j} \approx V_{A,j}$.  

268
Ideal Functionality 2 $\mathcal{F}_{\text{Ordered-MPC}}$

In the ideal model, a trusted third party $T$ is given the inputs, computes the functions $f, p$ on the inputs, and outputs to each player $i$ his output $y_i$ in the order prescribed by the ordering function. In addition, we model an ideal process adversary $S$ who attacks the protocol by corrupting players in the ideal setting.

Public parameters. $\lambda \in \mathbb{N}$, the security parameter; $n \in \mathbb{N}$, the number of parties; $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$, the function to compute; and $p : (\{0, 1\}^*)^n \rightarrow ([n] \rightarrow [n])$, the ordering function.

Private parameters. Each player $i \in [n]$ has input $x_i \in \{0, 1\}^*$. 

1. INPUT. Each player $i$ sends his input $x_i$ to $T$.
2. COMPUTATION. $T$ computes $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and $\pi = p(x_1, \ldots, x_n)$.
3. OUTPUT. The output proceeds in $n$ sequential output rounds. At the start of the $j$th round, $T$ sends the output value $\text{out}_{ij}$ to each party $i$, where $\text{out}_{ij} = y_{\pi(j)}$ and $\text{out}_{ij} = \bot$ for all $i \neq j$. When party $\pi(j)$ receives his output, he responds to $T$ with the message $\text{ack}$. (The players who receive $\bot$ are not expected to respond.) Upon receipt of the $\text{ack}$, $T$ proceeds to the $(j + 1)$th round — or, if $j = n$, then the protocol terminates.
4. OUTPUT OF VIEWS. At each output round, after receiving his message from $T$, each party produces an output, as follows. Each uncorrupted party $i$ outputs $y_i$ if he has already received his output, or $\bot$ if he has not. Each corrupted party outputs $\bot$. Additionally, the adversary $S$ outputs an arbitrary function of the information that he has learned during the execution of the ideal protocol. Let the output of party $i$ in the $j$th round be denoted by $V_{i,j}$, and let the view outputted by $S$ in the $j$th round be denoted by $V_{S,j}$. Let $V_{\text{ideal}}^{\text{Ordered-MPC}}$ denote the collection of all views for all output rounds:

$$V_{\text{ideal}}^{\text{Ordered-MPC}} = (\langle V_{S,1}, V_{1,1}, \ldots, V_{n,1} \rangle, \ldots, \langle V_{S,n}, V_{1,n}, \ldots, V_{n,n} \rangle).$$

(If the protocol is terminated early, then views for rounds which have not yet been started are taken to be $\bot$.)
In the context of ordered MPC, the standard guaranteed output delivery notion is insufficient. Instead, we define ordered output delivery, which requires in addition that all parties receive their outputs in the order prescribed by $p$.

**Definition 7.3.2** (Ordered output delivery). An ordered MPC protocol satisfies ordered output delivery if for any inputs $x_1, \ldots, x_n$, functionality $f$, and ordering function $p$, it holds that all parties receive their outputs before protocol termination, and moreover, if $\pi(i) < \pi(j)$, then party $i$ receives his output before party $j$ receives hers, where $\pi = p(x_1, \ldots, x_n)$.

We also define a natural relaxation of the fairness requirement for ordered MPC, called prefix-fairness. Although it is known that fairness is impossible for general functionalities in the presence of a dishonest majority, we show in the next subsection that prefix-fairness can be achieved even when a majority of parties are corrupt. We emphasize that this notion relaxes only the fairness requirement: that is, prefix-fair protocols satisfy full secrecy (and correctness) guarantees.

**Definition 7.3.3** (Prefix-fairness). An ordered MPC protocol is prefix-fair if for any inputs $x_1, \ldots, x_n$, it holds that the set of parties who have received their outputs at the time of protocol termination (or abortion) is a prefix of $(\pi(1), \ldots, \pi(n))$, where $\pi = p(x_1, \ldots, x_n)$ is the permutation induced by the inputs.

Prefix-fairness can be useful, for example, in settings where it is more important for one party to receive the output than the other; or where there is some prior knowledge about the trustworthiness of each party (so that more trustworthy parties may receive their outputs first).

### 7.3.2 Construction

Ordered MPC is achievable by using standard protocols for general MPC, as described in Protocol 1 below. The protocol has $n$ sequential output phases, so that the $n$ outputs can be issued in order. A subtle point is that because the ordering is a function of the input data, knowledge of the ordering may reveal information about
the input data. Thus, we have to “mask” the output values such that each party only
learns the minimal possible amount of information about the ordering: namely, his
own position in the ordering.

Protocol 1 Ordered MPC

Public parameters. \( \lambda \in \mathbb{N} \), the security parameter; \( n \in \mathbb{N} \), the number of parties;
\( k \in \mathbb{N} \), an upper bound on the number of corrupt parties; \( f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n \),
the function to be computed; and \( p : (\{0, 1\})^* \rightarrow ([n] \rightarrow [n]) \), the ordering function.

1. **Computing shares of \((\pi, y)\):** Using any general secure MPC protocol (such as [88]) on inputs \( x_1, \ldots, x_n \), jointly compute a \( k \)-out-of-\( n \) secret-sharing\(^{14}\) of \((\pi, y)\)
where \( y = (y_1, \ldots, y_n) = f(x_1, \ldots, x_n) \) and \( \pi = p(x_1, \ldots, x_n) \) is a permutation
of \([n]\). At the end of this step, each player possesses a share of the outputs
\( y = (y_1, \ldots, y_n) \) and of the permutation \( \pi \).

2. **Outputting \( y_1, \ldots, y_n \) in \( n \) phases:** In the \( i \)th output phase, player \( \pi^{-1}(i) \) will
learn his output. In phase \( i \) the parties run a new instance of a general secure
MPC protocol such that:

- Player \( j \)'s inputs to the protocol are: the shares of \( y \) and \( \pi \) that he got in
step 1, and a random string \( r_{i,j} \).
- The functionality computed is the following:

  for \( j \) from 1 to \( n \):
  
  if \( \pi(j) = i \) then \( z_{i,j} := y_j \oplus r_{i,j} \) else \( z_{i,j} = \bot \oplus r_{i,j} \).
  
  output \( z_i = (z_{i,1}, \ldots, z_{i,n}) \).

- To recover his output, each player \( j \) computes \( y'_{i,j} = z_{i,j} \oplus r_{i,j} \) for all \( i \). By
construction, there is exactly one \( i \in [n] \) for which \( y'_{i,j} \neq \bot \), and that is equal
to the output value \( y_j \) for player \( j \).

Check-points. There are \( n \) check-points. For \( i \in [n] \), the check-point \( C_i \) is at the
end of the \( i \)th output phase, when \( z_i \) is learned by all players.

In case of abort. When running the protocol for the honest majority setting,
the honest players continue until the end of the protocol regardless of other players’
behavior. When running the protocol for dishonest majority, if any party aborts in
an output phase\(^{15}\), then the honest players do not continue to the next phase.

In proving the security of Protocol 1, we refer to the security of modular compo-
sition of general protocols shown by [47], Theorem 5.

**Theorem 7.3.4.** Protocol 1 securely realizes \( F_{Ordered-MPC} \).

\(^{14}\)The standard definition of a secret-sharing scheme can be found in Chapter 2 (Preliminaries).

\(^{15}\)Each output phase consists of an execution of the underlying general MPC protocol. If a party
aborts at any time during (and before the end of) the execution of the underlying general MPC
protocol, this fact will be detected by all honest parties by the end of the phase.
Proof. Let $\rho_0$ denote the general MPC protocol execution in step 1, and let $\rho_i$ be the general MPC protocol execution in phase $i$ of step 2, for $i \in [n]$. For $j \in [n]$, let the protocol $\pi_j$ be the concatenation of the protocols $\rho_0, \ldots, \rho_j$. To prove security at each check-point, it is sufficient to prove that $\pi_j$ satisfies security for all $j \in [n]$: in other words, that the view outputted by any adversary in the real protocol execution at check-point $j$ can be simulated in the ideal execution. Finally, for all $j \in [n]$, the security of $\pi_j$ follows directly from the security of modular composition of general protocols ([47], Theorem 5).

Theorem 7.3.5. In the case of honest majority, Protocol 1 achieves fairness. In the dishonest majority setting, prefix-fairness is achieved.

Proof. Fairness holds in the honest majority case, since the honest players complete all output phases, and the shares that the honest players hold are sufficient to reconstruct each output $y_i$ (recall that the secret-sharing threshold $k$ is $\lceil n/2 \rceil$ in the honest majority case). In the dishonest majority setting, prefix-fairness holds since for all $i \in [n]$, all $n$ shares are required in order to reconstruct the output $y_{\pi(i)}$ in output phase $i$, and

- if the corrupt parties do not abort during the $i$th output phase, then by the security of Protocol 1, the output $y_{\pi(i)}$ associated with the $i$th output phase is delivered correctly to party $i$;
- if the corrupt parties abort during the $i$th output phase, then no outputs $y_{\pi(j)}$ for $j > i$ will be learned by any player, since the honest parties will not execute subsequent output phases.

7.4 Timed-delay MPC

In this section, we implementing time delays between different players receiving their outputs. The model is exactly as before, with $n$ players wishing to compute a function $f(x_1, \ldots, x_n)$ in an ordering prescribed by $p(x_1, \ldots, x_n)$ — except that now, there is an additional requirement of a delay after each player receives his output and before
the next player receives her output. To realize the timed-delay MPC functionality, we make use of time-lock and time-line puzzles, which are introduced in Sections 7.4.3 and 7.5.

7.4.1 Ideal functionality with time delays

We measure time delay in units of computation, rather than seconds of a clock: that is, rather than making any assumption about global clocks (or synchrony of local clocks)\(^\text{16}\), we measure time by the evaluations of a particular function (on random inputs), which we call the clock function.

For an algorithm A, let the run-time\(^\text{17}\) of A on input inp be denoted by time\(_A\)(inp). If A is probabilistic, the run-time will be a distribution over the random coins of A. Note that the exact run-time of an algorithm will depend on the underlying computational model in which the algorithm is run. In this work, all algorithms are assumed to be running in the same underlying computational model, and our definitions and results hold regardless of the specific computational model employed.

**Definition 7.4.1** (Security). A multi-party protocol F (with parameters λ, n, f, p, G) is said to securely realize \(\mathcal{F}_{\text{Timed-Delay-MPC}}\), if the following conditions hold.

1. The protocol description specifies n check-points \(C_1, \ldots, C_n\) corresponding to events during the execution of the protocol.
2. There exists a “clock function” \(g\) such that between any two consecutive checkpoints \(C_i, C_{i+1}\) during an execution of F, any one of the parties (in the real world) must

\(^{16}\)A particular issue that arises when considering a clock-based definition is that it is not clear that we can reasonably assume or prove that clocks are in synchrony between the real and ideal world — but this seems necessary in order to prove security by simulation in the ideal functionality.

We remark that if one is happy to assume the existence of a global clock (or synchrony of local clocks), then there are other ways to implement timed-delay MPC which sidestep many of the issues inherent in the arguably more realistic model where clocks may not be perfectly synchronized between different (adversarial) parties. One example is the “Bitcoin model” where the assumption is that the Bitcoin block-chain can serve as a global clock: in this model, existing protocols such as [35] implement some time-delays in MPC, and it seems likely that such protocols can be adapted to achieve our notion of timed-delay MPC.

\(^{17}\)The use of checkpoints is introduced to capture the views of players and the adversary at intermediate points in protocol execution.

---

273
Ideal Functionality 3 $F_{\text{Timed-Delay-MPC}}$

In the ideal model, a trusted third party $T$ is given the inputs, computes the functions $f, p$ on the inputs, and outputs to each player $i$ his output $y_i$ in the order prescribed by the ordering function. Moreover, $T$ imposes delays between the issuance of one party's output and the next. In addition, we model an ideal process adversary $S$ who attacks the protocol by corrupting players in the ideal setting.

**Public parameters.** $\lambda \in \mathbb{N}$, the security parameter; $n \in \mathbb{N}$, the number of parties; $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$, the function to be computed; $p : (\{0, 1\}^*)^n \rightarrow ([n] \rightarrow [n])$, the ordering function; and $G = G(\lambda) \in \mathbb{N}$, the number of time-steps between the issuance of one party's output and the next.

**Private parameters.** Each player $i \in [n]$ has input $x_i \in \{0, 1\}^*$.

1. **INPUT.** Each player $i$ sends his input $x_i$ to $T$. If, instead of sending his input, any player sends the message quit, then the computation is aborted.

2. **COMPUTATION.** $T$ computes $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and $\pi = p(x_1, \ldots, x_n)$.

3. **OUTPUT.** The output proceeds in $n$ sequential output phases. At each phase $j$, $T$ waits for $G$ time-steps, then sends the $j$th output, $y_{\pi(j)}$, to party $\pi(j)$.

4. **OUTPUT OF VIEWS.** At the end of each output phase, each party produces an output as follows. Each uncorrupted party $i$ outputs $y_i$ as his view if he has already received his output, or $\perp$ if he has not. Each corrupted party outputs $\perp$. Additionally, the adversary $S$ outputs an arbitrary function of the information that he has learned during the execution of the ideal protocol, after each checkpoint.

Let the output of party $i$ in the $j$th round be denoted by $\mathcal{V}_{i,j}$, and let the view outputted by $S$ in the $j$th round be denoted by $\mathcal{V}_{S,j}$. Let $\mathcal{V}_{\text{ideal}}^{\text{Timed-Delay-MPC}}$ denote the collection of all views for all output phases:

$$\mathcal{V}_{\text{ideal}}^{\text{Timed-Delay-MPC}} = ((\mathcal{V}_{S,1}, \mathcal{V}_{1,1}, \ldots, \mathcal{V}_{n,1}), \ldots, (\mathcal{V}_{S,n}, \mathcal{V}_{1,n}, \ldots, \mathcal{V}_{n,n})).$$
be able to locally run $\Omega(G)$ sequential evaluations of $g$ on random inputs. $g$ may also be a protocol (involving $n' \leq n$ parties) rather than a function, in which case we instead require that any subset consisting of $n'$ parties must be able to run $\Omega(G)$ sequential evaluations of $g$ (on random inputs) over the communication network being used for the main multi-party protocol $F$. Then, we say that $F$ is “clocked by $g$.”

3. Take any PPT adversary $A$ attacking the protocol $F$ by corrupting a subset of players $S \subset [n]$, which outputs an arbitrary function $V_{A,j}$ of the information that it has learned in the protocol execution after each check-point $C_j$. Let

$$V_{A}^{\text{real}} = ((V_{A,1}, V_{1,1}, \ldots, V_{n,1}), \ldots, (V_{A,n}, V_{1,n}, \ldots, V_{n,n}))$$

be the tuple consisting of the adversary $A$’s outputted views along with the views of the real-world parties as specified in the ideal functionality description. Then there is a PPT ideal adversary $S$ which, attacking $\mathcal{F}_{\text{Timed-Delay-MPC}}$ by corrupting the same subset $S$ of players, can output views $V_{S,1}, \ldots, V_{S,n}$ (at check-points $C_1, \ldots, C_n$ respectively) such that for each $j \in [n]$, it holds that

$$|\Pr[D(V_{S,j}, V_{1,j}, \ldots, V_{n,j}) = 1] - \Pr[D(V_{A,j}, V_{1,j}, \ldots, V_{n,j}) = 1]| \leq \text{negl}(\lambda),$$

for any distinguisher $D$ such that

$$\Pr_{v \leftarrow V}[\text{time}_D(v) \leq j \cdot \text{time}_G()] = 1/\text{poly}(\lambda),$$

when $V$ is the distribution of views outputted by $A$ or $S$ (that is, for $V \in \{(V_{S,j}, V_{1,j}, \ldots, V_{n,j}), (V_{A,j}, V_{1,j}, \ldots, V_{n,j})\}$), and $G$ is the algorithm that computes the function $g$ sequentially on $G$ random inputs.

### 7.4.2 Realizing timed-delay MPC with dummy rounds

A simple protocol for securely realizing timed-delay MPC is to implement delays by running $G$ “dummy rounds” of communication in between issuing outputs to different
players.

**Theorem 7.4.2.** In the presence of honest majority, Protocol 2 securely realizes $\mathcal{T}_{\text{Timed-Delay-MPC}}$ clocked by $g_{\text{dum}}$.

**Proof.** Let $A$ be any PPT adversary attacking Protocol 2 by corrupting a subset of players $S \subset [n]$, and let

$$V_A^{\text{real}} = ((V_{A,1}, V_{1,1}, \ldots, V_{n,1}), \ldots, (V_{A,n}, V_{1,n}, \ldots, V_{n,n}))$$

be the tuple consisting of the adversary $A$'s outputted views along with the views of the real-world parties (as specified in the description of $\mathcal{T}_{\text{Timed-Delay-MPC}}$). In order to show that condition 3 of the security definition (Definition 7.4.1) holds, we need to show that there is a PPT ideal adversary $S$ which, given access to $\mathcal{T}_{\text{Timed-Delay-MPC}}$ and corrupting the same subset $S$ of players, can output views $V_{S,j}^*$ such that $V_A^{\text{real}} \approx V_{S,j}^*$.

Recall that the adversary’s view can be any function of the inputs of the corrupt parties and the messages that the corrupt parties see during the protocol execution. In particular, it is sufficient to show that there is an ideal adversary $S$ which can output views $V_{S,j}^*$ which are indistinguishable from the transcript of all the messages that the corrupt parties see during the real protocol execution.

Protocol 2 consists of sequential executions of the underlying general MPC protocol and the mini-protocol $g_{\text{dum}}$. When the mini-protocol executions are removed from Protocol 2, the resulting protocol is identical to Protocol 1. Hence, by Theorem 7.3.4, there is an ideal adversary $S'$ which, given access to $\mathcal{T}_{\text{Timed-Delay-MPC}}$ and corrupting the same subset $S$ of players, can output views $V_{S',j}^*$ such that which are indistinguishable from the transcript of all the messages that the corrupt parties see during the protocol execution.

---

18 For the honest majority setting, we set $k = \lceil n/2 \rceil$. For the dishonest majority setting, $k = n$.
19 Each output phase consists of an execution of the underlying general MPC protocol preceded by $G$ dummy rounds. If a party aborts before the completion of the $G$ dummy rounds, this fact will be detected by all parties in the dummy round in which the abort happens, because every party is supposed to communicate with every other party in each dummy round. If a party aborts at any time during (and before the end of) the execution of the underlying general MPC protocol, this fact will be detected by all honest parties by the end of the phase.
Protocol 2 Timed-delay MPC with dummy rounds

Public parameters. \( \lambda \in \mathbb{N} \), the security parameter; \( n \in \mathbb{N} \), the number of parties; \( f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n \), the function to be computed; \( p : (\{0, 1\}^*)^n \rightarrow ([n] \rightarrow [n]) \), the ordering function; and \( G \in \text{poly}(\lambda) \), the number of time-steps between the issuance of one party's output and the next.

1. Computing shares of \((\pi, y)\): Using any general secure MPC protocol (such as [88]), jointly compute an \( k \)-out-of-\( n \) secret-sharing of \((\pi, y)\) where \( y = (y_1, \ldots, y_n) = f(x_1, \ldots, x_n) \) and permutation \( \pi = p(x_1, \ldots, x_n) \) on the players' inputs. At the end of this step, each player possesses a share of the outputs \( y = (y_1, \ldots, y_n) \) and of the permutation \( \pi \).

2. Outputting \( y_1, \ldots, y_n \) in \( n \) phases: The outputs will occur in \( n \) phases: in the \( i \)th phase, player \( \pi^{-1}(i) \) will learn his output. In each phase, the players first run \( G \) "dummy rounds" of communication. A dummy round is a "mini-protocol" defined as follows (let this mini-protocol be denoted by \( \text{gdum} \)):
   - each player initially sends the message challenge to every other player;
   - each player responds to each challenge he receives with a message response.

In each phase, after the dummy rounds have been completed, the parties will run a new instance of a general secure MPC protocol. In phase \( i \):
   - Player \( j \)'s inputs to the protocol are: the shares of \( y \) and \( \pi \) that he got in step 1, and a fresh random string \( r_{i,j} \).
   - The functionality computed in each phase \( i \in [n] \) is:
     - for \( j \) from 1 to \( n \):
       - if \( \pi(j) = i \) then \( z_{i,j} := y_j \oplus r_{i,j} \) else \( z_{i,j} = \perp \oplus r_{i,j} \).
       - output \( z_i = (z_{i,1}, \ldots, z_{i,n}) \).
   - To recover his output, each player \( j \) computes \( y'_{i,j} = z_{i,j} \oplus r_{i,j} \) for all \( i \). By construction, there is exactly one \( i \in [n] \) for which \( y'_{i,j} \neq \perp \), and that is equal to the output value \( y_j \) for player \( j \).

Check-points. There are \( n \) check-points. For \( i \in [n] \), the check-point \( C_i \) is at the end of the \( i \)th output phase, when \( z_i \) is learned by all players.

In case of abort. When running the protocol for the honest majority setting, the honest players continue until the end of the protocol regardless of other players' behavior. When running the protocol for dishonest majority, if any party aborts in an output phase\(^{19}\), then the honest players do not continue to the next phase.
during the $n+1$ executions of the underlying general MPC protocol within Protocol 2. The only other messages that are sent in Protocol 2 are the “dummy” messages challenge and response, which are fixed messages that do not depend on the players’ inputs. In fact, the transcript of an execution of $g_{dum}$ is a deterministic sequence of challenge and response. It follows that there exists an ideal adversary $S$ which, by calling $S'$ and adding the deterministic transcript corresponding to each execution of $g_{dum}$, can output views $\mathcal{V}_{S,j}$ which are indistinguishable from the transcript of all the messages that the corrupt parties see during the real execution of Protocol 2.

Finally, it remains to show that condition 2 of the security definition (Definition 7.4.1) is satisfied. The players are literally running $g$ over the MPC network $G$ times in between issuing outputs, so it is clear that condition 2 holds.

One downside of the simple solution above is that it requires all (honest) parties to be online and communicating until the last player receives his output. To address this, in Section 7.4.3 we propose an alternative solution based on timed-release cryptography, at the cost of an additional assumption that all players have comparable computing speed (within a logarithmic factor).

7.4.3 Realizing timed-delay MPC with time-lock puzzles

Informally, a time-lock puzzle is a primitive which allows “locking” of data, such that it will be released after a pre-specified time delay, and no earlier. Our next protocol, instead of issuing outputs to players in the clear, gives to each party his output locked into a time-lock puzzle; and in order to enforce the desired ordering, the delays required to unlock the puzzles are set to be an increasing sequence. We first give the definition of time-lock puzzles (in Section 7.4.3) then describe and prove security of our time-lock-based protocol (in Section 7.4.3).

Time-lock puzzles

The delayed release of data in MPC protocols can be closely linked to the problem of “timed-release crypto” in general, which was introduced by [133] and constructed
first by [155] with their proposal of time-lock puzzles. We assume time-lock puzzles with a particular structure (that is present in all known implementations): namely, the passage of "time" will be measured by sequential evaluations of a function (TimeStep). Unlocking a t-step time-lock puzzle can be considered analogous to following a chain of t pointers, at the end of which there is a special value $x_t$ (e.g., a decryption key) that allows retrieval of the locked data.

Definition 7.4.3 (Time-lock puzzle scheme). A time-lock puzzle scheme is a tuple of PPT algorithms $T = (\text{Lock}, \text{TimeStep}, \text{Unlock})$ as follows:

- $\text{Lock}(1^\lambda, d, t)$ takes parameters $\lambda \in \mathbb{N}$ the security parameter, $d \in \{0, 1\}^\ell$ the data to be locked, and $t \in \mathbb{N}$ the number of steps needed to unlock the puzzle, and outputs a time-lock puzzle $P = (x, t, b, a) \in \{0, 1\}^\ell \times \mathbb{N} \times \{0, 1\}^{n'} \times \{0, 1\}^{n''}$ where $\ell, n, n', n'' \in \text{poly}(\lambda)$.
- $\text{TimeStep}(1^\lambda, x', a')$ takes parameters $\lambda \in \mathbb{N}$ the security parameter, a bit-string $x' \in \{0, 1\}^n$, and auxiliary information $a'$, and outputs a bit-string $x'' \in \{0, 1\}^n$.
- $\text{Unlock}(1^\lambda, x', b')$ takes parameters $\lambda \in \mathbb{N}$ the security parameter, a bit-string $x' \in \{0, 1\}^n$, and auxiliary information $b' \in \{0, 1\}^{n'}$, and outputs some data $d' \in \{0, 1\}^{\ell}$.

To unclutter notation, we will sometimes omit the initial security parameter of these functions (writing, e.g., simply $\text{Lock}(d, t)$). We now define some auxiliary functions. For a time-lock puzzle scheme $T = (\text{Lock}, \text{TimeStep}, \text{Unlock})$ and $i \in \mathbb{N}$, let $\text{IterateTimeStep}_T^i$ denote the following function:

$$\text{IterateTimeStep}_T^i(i, x, a) = \text{TimeStep}(\text{TimeStep}(...(\text{TimeStep}(x, a), a)...), a).$$
Define $\text{CompleteUnlock}^T$ to be the following function:

$$
\text{CompleteUnlock}^T((x, t, b, a)) = \text{ Unlock}(\text{IterateTimeStep}^T(t, x, a), b),
$$

that is, the function that should be used to unlock a time-lock puzzle outputted by Lock.

The following definitions formalize correctness and security for time-lock puzzle schemes.

**Definition 7.4.4** (Correctness). A time-lock puzzle scheme $T = (\text{Lock}, \text{TimeStep}, \text{Unlock})$ is correct if the following holds (where $\lambda$ is the security parameter):

$$\Pr_{(x, t, b, a) \leftarrow \text{Lock}(d, t)} \left[ \text{CompleteUnlock}^T((x, t, b, a)) \neq d \right] \leq \text{negl}(\lambda).$$

**Definition 7.4.5** (Security). Let $T = (\text{Lock}, \text{TimeStep}, \text{Unlock})$ be a time-lock puzzle scheme. $T$ is secure if it holds that: for all $d, d' \in \{0, 1\}^t, t \in \text{poly}(\lambda)$, if there exists an adversary $A$ that solves the time-lock puzzle $\text{Lock}(d, t)$, that is,

$$\Pr_{P \leftarrow \text{Lock}(d, t)} [A(P) = d] = \varepsilon \text{ for some non-negligible } \varepsilon,$$

then for each $j \in [t]$, there exists an adversary $A_j$ such that

$$\Pr_{P' \leftarrow \text{Lock}(d', j)} [A_j(P') = d'] \geq 1 - \text{negl}(\lambda), \text{ and}
\Pr_{P \leftarrow \text{Lock}(d, t), P' \leftarrow \text{Lock}(d', j)} [\text{time}_A(P) \geq (t/j) \cdot \text{time}_{A_j}(P') \mid A(P) = d] \geq 1 - \text{negl}(\lambda).$$

**Protocol based on time-lock puzzles**

Because of the use of time-lock puzzles by different parties in the protocol that follows, we require an additional assumption that all players have comparable computing power (within a logarithmic factor).
Relative-Delay Assumption. The difference in speed of performing computations between any two parties \(i, j \in [n]\) is at most a factor of \(B = O(\log(\lambda))\).

Protocol 3 Timed-delay MPC with time-lock puzzles

Public parameters. \(\lambda \in \mathbb{N}\), the security parameter; \(n \in \mathbb{N}\), the number of parties; \(f : (\{0, 1\}^*)^n \to (\{0, 1\}^*)^n\), the function to be computed; \(p : (\{0, 1\}^*) \rightarrow ([n] \to [n])\), the ordering function; \(B = O(\log(\lambda))\), the maximum factor of difference between any two parties' computing power; \(G \in \text{poly}(\lambda)\), the number of time-steps between the issuance of one party's output and the next; and \(T = \{\text{Lock}, \text{TimeStep}, \text{Unlock}\}\) a time-lock puzzle scheme.

Inputs. Each party \(i\) has input \(x_i\).

Protocol steps. Let \((y_1, \ldots, y_n) = f(x_1, \ldots, x_n)\) and \(\pi = p(x_1, \ldots, x_n)\). Define \(t_1 = 1\) and \(t_{i+1} = (B \cdot G + 1) \cdot t_i\) for \(i \in [n-1]\). Compute \((P_1, \ldots, P_n)\), where each \(P_i = (x_i, t_{\pi(i)}, a_i, b_i)\) is a time-lock puzzle computed as
\[
P_i = \text{Lock}(y_i \oplus r_i, t_{\pi(i)})
\]
where each \(r_i\) is a random string provided as input randomness by party \(i\).

Outputs. For each \(i \in [n]\), the puzzle \(P_i\) is outputted to party \(i\). The players all receive their respective outputs at the same time, then recovers his output \(y_i\) by solving his time-lock puzzle, and finally “unmasking” the result by XORing with his random input \(r_i\).

Check-points. There are \(n\) check-points. For \(i \in [n]\), the check-point \(C_i\) is the event of party \(\pi(i)\) learning his eventual output \(y_{\pi(i)}\) (i.e., when he finishes solving his time-lock puzzle).

For the following theorem, we assume that each player \(i\) uses the optimal algorithm to solve his puzzle \(P_i\) that outputs the correct answer. Without this assumption, any further protocol analysis would not make sense: there can always be a “lazy” player who willfully uses a very slow algorithm to solve his puzzle, who will as a result learn his eventual output much later in the order than he could otherwise have done.

The property that we aim to achieve is that every player could learn his output at his assigned position in the ordering \(\pi\), with appropriate delays before and after he learns his output.

Theorem 7.4.6. Suppose that the Relative-Delay Assumption holds, and each player \(i\) uses the optimal algorithm to solve his puzzle \(P_i\) that outputs (with overwhelming probability) the correct answer. Then, Protocol 3 securely realizes \(\mathcal{F}_{\text{Timed-Delay-MPC}}\)
when there is an honest majority.

**Proof.** First, we prove that condition 2 of the security definition (Definition 7.4.1) is satisfied. Let $A_i$ denote the algorithm that party $i$ uses to solve his time-lock puzzle, and let the time at which party $i$ learns his answer $y_i$ be denoted by $\tau_i = \text{time}_{A_i}(P_i)$. By the security of the time-lock puzzles, there exists an algorithm $A'_i$ that player $i$ could use to solve the puzzle $\text{Lock}(0^t, 1)$ in time $\tau_i/t_i$. Moreover, by the Relative-Delay Assumption, it holds that no player can solve the puzzle $\text{Lock}(0^t, 1)$ more than $B$ times faster than another player: that is, $\max_i(\tau_i/t_i) \leq B \cdot \min_i(\tau_i/t_i)$. It follows that even the slowest player (call him $i^*$) would be able to run $t_i/B$ executions of $A'_i$ within time $\tau_i$, for any $i$.

Without loss of generality, assume that the ordering function $p$ is the identity function. Consider any consecutive pair of checkpoints $C_i, C_{i+1}$. These checkpoints occur at times $\tau_i$ and $\tau_{i+1}$, by definition. We have established that in time $\tau_i$, player $i^*$ can run $t_i/B$ executions of $A'_i$, and in time $\tau_{i+1}$, he can run $t_{i+1}/B$ executions of $A'_i$. It follows that in between the two checkpoints (i.e., in time $\tau_{i+1} - \tau_i$), he can run $(t_{i+1} - t_i)/B$ executions of $A'_i$. Substituting in the equation $t_{i+1} = (B \cdot G + 1) \cdot t_i$ from the protocol definition, we get that player $i^*$ can run $G \cdot t_i$ executions of $A'_i$ between checkpoints $C_i$ and $C_{i+1}$. Since $t_i \geq 1$ for all $i$, this means that $i^*$ can run at least $G$ executions of $A'_i$ between any consecutive pair of checkpoints. Hence, condition 2 holds.

We now prove condition 3. Let $G$ be the algorithm that evaluates $A'_i$ sequentially $G$ times on random inputs. It is sufficient to show that for any adversary $A$ attacking the protocol by corrupting a subset $S \subset [n]$ of players, which outputs a view $V_{A,j}$ at each checkpoint $j$ which is the *transcript of all messages that it has seen so far*, there is an ideal adversary $S$ which outputs views $V_{S,1}, \ldots, V_{S,n}$ such that for any $j \in [n]$, for any distinguisher $D$ whose run-time satisfies the conditions in Definition 7.4.1, item 3,

$$|\Pr[D(V_{S,j}, V_{1,j}, \ldots, V_{n,j}) = 1] - \Pr[D(V_{A,j}, V_{1,j}, \ldots, V_{n,j}) = 1]| \leq \text{negl}(\lambda).$$
Recall that there are $n$ sequential output stages in the ideal functionality $\mathcal{F}_{\text{Timed-Delay-MPC}}$. Consider an ideal adversary $S$ attacking $\mathcal{F}_{\text{Timed-Delay-MPC}}$ by corrupt a set of parties $S \subseteq [n]$. Let $\vec{inp}$ denote the vector of inputs and input randomness of the corrupt parties (note that these are known to $S$). Take any $i \in [n]$. In the ideal protocol execution, $S$ learns the following in the $\pi(i)$th output stage:

- nothing, if $i \notin S$; or
- the input value $x_i$, the input randomness $r_i$, and the eventual output $y_i$ if $i \in S$.

Note that as a result, $S$ learns $\pi(i)$ at output stage $\pi(i)$, for each $i \in S$. The delay values $t_1, \ldots, t_n$ are a fixed sequence of values independent of the parties' inputs, so they are known to $S$. Thus, at each check-point $j \subseteq [n]$, the ideal adversary $S$ can compute $n$ time-lock puzzles:

$$\hat{P}_{j,i} = \begin{cases} 
\text{Lock}(y_i \oplus r_i, t_{\pi(i)}) & \text{if } 1 \leq i \leq j \\
\text{Lock}(r_i, t_{\pi(i)}) & \text{if } j < i \leq n
\end{cases}$$

Let the ideal adversary $S$ output the following view at each check-point $j$:

$$\mathcal{V}_{S,j} = S_j(\vec{inp}, (\hat{P}_{j,1}, \ldots, \hat{P}_{j,n})),$$

where $S_j$ is the ideal adversary (for the underlying general MPC protocol) that simulates the adversary's $j$th view $\mathcal{V}_{A,j}$.

We now analyze the distribution of the puzzles $\hat{P}_{j,i}$. For the range $1 \leq i \leq j$, the puzzle $\hat{P}_{j,i}$ is by definition identically distributed to the puzzle that is outputted to player $i$ in the real execution of Protocol 3. Now take any $j \subseteq [n]$, and let $D$ be any distinguisher whose run-time satisfies the conditions in Definition 7.4.1, item 3. Recall that the players are assumed to solve the time-lock puzzles using the optimal algorithm. Hence, it follows from the security of the underlying time-lock puzzle scheme that for any $i$ in the range $j < i \leq n$,

$$\left| \Pr[D(P_{j,i}) = 1] - \Pr[D(\hat{P}_{j,i}) = 1] \right| \leq \text{negl}(\lambda).$$
Since we defined the outputs of $S$ to be $\mathcal{V}_{S,j} = S_j(\text{inp}, (\tilde{P}_{j,1}, \ldots, \tilde{P}_{j,n}))$ for $j \in [n]$, it follows that

$$\left| \Pr[D(\mathcal{V}_{S,j}, \mathcal{V}_{1,j}, \ldots, \mathcal{V}_{n,j}) = 1] - \Pr[D(\mathcal{V}_{A,j}, \mathcal{V}_{1,j}, \ldots, \mathcal{V}_{n,j}) = 1] \right| \leq \negl(\lambda)$$

as required. We conclude that Protocol 3 securely realizes $\mathcal{T}_{\text{Timed-Delay-MPC}}$ clocked by $A'$.

A few remarks are in order. In Protocol 3, all the parties can stop interacting as soon as all the puzzles are outputted. When the locking algorithm $\text{Lock}(d, t)$ has run-time that is independent of the delay $t$, the run-time of Protocol 3 is also independent of the delay parameters. (This is achievable using the [155] time-lock construction, for example.) Alternatively, using a single time-line puzzle in place of the time-lock puzzles in Protocol 3 can improve efficiency, since the time required to generate a time-line puzzle is dependent only on the longest delay $t_n$, whereas the time required to generate $n$ separate time-lock puzzles depends on the sum of all the delays, $t_1 + \cdots + t_n$.

### 7.5 Time-line puzzles

We now introduce the more general, novel definition of time-line puzzles, which can be useful for locking together many data items with different delays for a single recipient, or for locking data for a group of people. In the latter case, it becomes a concern that computation speed will vary between parties: indeed, the scheme will be unworkable if some parties have orders of magnitude more computing power than others, so some assumption is required on the similarity of computing power among parties, such as the Relative-Delay Assumption of Section 7.4.3. When a time-line puzzle is given to a single recipient, then no additional assumptions are required.

We remark that time-line puzzles could be used (instead of a set of time-lock puzzles) to realize Protocol 3. More generally, we present this new notion because we believe that time-line puzzles may be of independent interest as a timed-release
In some ways, a time-line puzzle can be thought of as a primitive that packages a sequence of time-lock puzzles together into a unified system about which we can reason and give security guarantees. However, time-line puzzles can also provide concrete advantages over a collection of time-lock puzzles. For example, when issuing many time-lock puzzles to one recipient, the recipient has to run the computation for all of the puzzles in parallel: that is, he does $O(m \cdot t)$ computation where $m$ is the number of data items and $t$ is the time-delay. If instead he gets a time-line puzzle, he only has to run one puzzle's worth of computation in order to unlock all the data items: that is, he does only $O(t)$ computation, just like for a single time-lock puzzle.

**Definition 7.5.1** (Time-line puzzles). A time-line puzzle scheme is a family of PPT algorithms $\mathcal{T} = \{ \text{Lock}_m, \text{TimeStep}_m, \text{Unlock}_m \}_{m \in \mathbb{N}}$ as follows:

- $\text{Lock}_m(1^\lambda, (d_1, \ldots, d_m), (t_1, \ldots, t_m))$ takes parameters $\lambda \in \mathbb{N}$ the security parameter, $(d_1, \ldots, d_m) \in \{0, 1\}^m$ the data items to be locked, and $(t_1, \ldots, t_m) \in \mathbb{N}^m$ the number of steps needed to unlock each data item (respectively), and outputs a puzzle

$$P = (x, (t_1, \ldots, t_m), (b_1, \ldots, b_m), a) \in \{0, 1\}^n \times \mathbb{N} \times (\{0, 1\}^n)^m \times \{0, 1\}^n$$

where $n, n', n'' \in \text{poly}(\lambda)$, and $a$ can be thought of as auxiliary information.

- $\text{TimeStep}_m(1^\lambda, x', a')$ takes parameters $\lambda \in \mathbb{N}$ the security parameter, a bit-string $x' \in \{0, 1\}^n$, and auxiliary information $a'$, and outputs a bit-string $x'' \in \{0, 1\}^n$.

- $\text{Unlock}_m(1^\lambda, x', b')$ takes parameters $\lambda \in \mathbb{N}$ the security parameter, a bit-string $x' \in \{0, 1\}^n$, and auxiliary information $b' \in \{0, 1\}^n$, and outputs some data $d' \in \{0, 1\}^\ell$.

In terms of the “pointer chain” analogy above, solving a time-line puzzle may be thought of as following a pointer chain where not one but many keys are placed along the chain, at different locations $t_1, \ldots, t_m$. Each key $x_{t_i}$ in the pointer chain depicted below enables the “unlocking” of the locked data $b_i$: for example, $b_i$ could be the encryption of the $i$th data item $d_i$ under the key $x_{t_i}$.
Using similar notation to that defined for time-lock puzzles: for a time-line puzzle scheme $\mathcal{T}$, let $\text{IterateTimeStep}^\mathcal{T}_m$ denote the following function:

$$\text{IterateTimeStep}^\mathcal{T}_m(i, x, a) = \text{TimeStep}_m(\text{TimeStep}_m(\ldots(\text{TimeStep}_m(x, a), a), a)\ldots), a) .$$

Define $\text{CompleteUnlock}^\mathcal{T}_{m,i}$ to be the following function:

$$\text{CompleteUnlock}^\mathcal{T}_{m,i}((x, t_i, b_i, a)) = \text{Unlock}_m(\text{IterateTimeStep}^\mathcal{T}_m(t_i, x, a), b_i) ,$$

that is, the function that should be used to unlock the $i$th piece of data locked by a time-line puzzle which was generated by $\text{Lock}_m$. We now define correctness and security for time-line puzzle schemes.

**Definition 7.5.2 (Correctness).** A time-line puzzle scheme $\mathcal{T}$ is correct if for all $m \in \text{poly}(\lambda)$ and for all $i \in [m]$, it holds that

$$\Pr_{(x, t_i, b_i, a) \leftarrow \text{Lock}_m(d^\mathcal{T}_i)} \left[ \text{CompleteUnlock}^\mathcal{T}_{m,i}((x, t_i, b_i, a)) \neq d_i \right] \leq \text{negl}(\lambda) ,$$

where $\lambda$ is the security parameter, $d^\mathcal{T}_i = (d_1, \ldots, d_m)$, and $i^\mathcal{T} = (t_1, \ldots, t_m)$.

Security for time-line puzzles involves more stringent requirements than security for time-lock puzzles. We define security in terms of two properties which must be satisfied: timing and hiding. The timing property is very similar to the security requirement for time-lock puzzles, and gives a guarantee about the relative amounts of time required to solve different time-lock puzzles. The hiding property ensures (informally speaking) that the ability to unlock any given data item that is locked in a time-line puzzle does not imply the ability to unlock any others. The security definition (Definition 7.5.3, below) refers to the following security experiment.
The experiment $\text{HidingExp}_{A, T}(\lambda)$

1. $A$ outputs $m \in \text{poly}(\lambda)$ and data vectors $\vec{d}_0, \vec{d}_1 \in \{0, 1\}^\ell m$ and a time-delay vector $\vec{t} \in \mathbb{N}^m$.
2. The challenger samples $(\beta_1, \ldots, \beta_m) \leftarrow \{0, 1\}^m$, computes the time-line puzzle $(x, \vec{t}, \vec{b}, a) = \text{Lock}_m(1^\lambda, ((d_{\beta_1}), \ldots, (d_{\beta_m})_m), \vec{t})$, and sends $(x, a)$ to $A$.
3. $A$ sends a query $i \in [m]$ to the challenger. The challenger responds by sending $b_i$ to $A$. This step may be repeated up to $m - 1$ times. Let $I$ denote the set of queries made by $A$.
4. $A$ outputs $i' \in [m]$ and $\beta' \in \{0, 1\}$.
5. The output of the experiment is 1 if $i' \notin I$ and $\beta' = \beta_i$. Otherwise, the output is 0.

**Definition 7.5.3 (Security).** Let $T = \{(\text{Lock}_m, \text{TimeStep}_m, \text{Unlock}_m)\}_{m \in \mathbb{N}}$ be a time-line puzzle scheme. $T$ is secure if it satisfies the following two properties.

- **TIMING:** For all $m \in \text{poly}(\lambda)$ and $\vec{d}_0, \vec{d}_1 \in \{0, 1\}^\ell m$ and $\vec{t} = (t_1, \ldots, t_m)$, if there exists an adversary $A$ that solves any one of the puzzles defined by the time-line, that is,

$$\Pr_{P \leftarrow \text{Lock}_m(\vec{d}, \vec{t})}[A(P) = d_i] = \varepsilon$$

for some non-negligible $\varepsilon$ and some $i \in [m]$,

then for all $j \in [t_i]$ and all $\vec{t}' \in [m]^m$, there exists an adversary $A_{j, \vec{t}'}$ such that

$$\Pr_{P' \leftarrow \text{Lock}_m(\vec{d}, \vec{t}')}[A_{j, \vec{t}'}(P') = d_j] \geq 1 - \text{negl}(\lambda), \text{ and}$$

$$\Pr[\text{time}_A(P) \geq (t'_j/t_i) \cdot \text{time}_{A_{j, \vec{t}'}}(P') \mid A(\text{Lock}_m(\vec{d}, \vec{t})) = d_i] \geq 1 - \text{negl}(\lambda).$$

- **HIDING:** For all PPT adversaries $A$, it holds that

$$\Pr[\text{HidingExp}_{A, T}(\lambda) = 1] \leq 1/2 + \text{negl}(\lambda).$$
Constructions of time-line puzzles. In the following subsections, we describe and prove the security of two constructions of time-line puzzle schemes. One of these schemes is based on a concrete assumption (specifically, on the sequentiality of modular exponentiation, like the time-lock puzzles of [155]), whereas the other is based on the existence of a “black-box” inherently sequential hash function.

### 7.5.1 Black-box construction from inherently sequential hash functions

**Definition 7.5.4 (Inherently-sequential hash function).** Let \( \mathcal{H} = \{ H_\lambda \}_{\lambda \in \mathbb{N}} \) be a family of functions where each \( H_\lambda = \{ h_s : \{0,1\}^\lambda \rightarrow \{0,1\}^{k \in \{0,1\}^n} \) for some \( n \in \text{poly}(\lambda) \). Suppose that evaluating \( h_s(r) \) for \( r \leftarrow \{0,1\}^\lambda \) takes time \( \Omega(T) \). \( \mathcal{H}_\lambda \) is said to be inherently sequential if evaluating \( h_s'(r) \) for \( s \leftarrow \{0,1\}^n, r \leftarrow \{0,1\}^\lambda \) takes time \( \Omega(t \cdot T) \), and the output of \( h_s'(r) \) is pseudorandom.

The time-line puzzle construction in this section relies on the following assumption about the existence of inherently sequential functions.

**Assumption 7.5.5.** There exists a family of functions \( \tilde{\mathcal{H}} = \{ \tilde{H}_\lambda \}_{\lambda \in \mathbb{N}} \) which is inherently sequential. For the sake of defining syntax, for \( \lambda \in \mathbb{N} \), let

\[
\tilde{H}_\lambda = \{ \tilde{h}_s : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda_s} \}_{s \in \{0,1\}^n}.
\]

**Definition 7.5.6.** BB-TimeLinePuzzle is a time-line puzzle defined as follows, where \( \tilde{\mathcal{H}}_\lambda \) is the inherently sequential hash function family from Assumption 7.5.5:

- **Lock** \( m(1^\lambda, (d_1, \ldots, d_m), (t_1, \ldots, t_m)) \) takes input data \( (d_1, \ldots, d_m) \in \{0,1\}^\lambda \), samples random values \( s \leftarrow \{0,1\}^n, x \leftarrow \{0,1\}^\lambda \), and outputs the puzzle

\[
P = (x, (t_1, \ldots, t_m), s, (d_1 \oplus \tilde{h}_s^{t_1}(x), \ldots, d_m \oplus \tilde{h}_s^{t_m}(x))).
\]

- **TimeStep** \( m(1^\lambda, i, x', a') \) outputs \( \tilde{h}_{a'}(x') \).
- **Unlock** \( m(1^\lambda, x', b') \) outputs \( x' \oplus b' \).
It is clear that **BB-TimeLinePuzzle** satisfies correctness, so we proceed to prove security.

**Theorem 7.5.7.** If Assumption 7.5.5 holds, then **BB-TimeLinePuzzle** is a secure time-line puzzle.

**Proof.** Given a time-line puzzle, in order to correctly output a piece of locked data \(d_i\), the adversary \(A\) must compute the associated mask \(\widetilde{h}^i_s(x)\). This is because

- all components of the puzzle apart from the masked value \(d_i \oplus \widetilde{h}^i_s(x)\) are independent of the locked data \(d_i\), and
- the mask \(\widetilde{h}^i_s(x)\) is pseudorandom (by Assumption 7.5.5), so the masked value \(d_i \oplus \widetilde{h}^i_s(x)\) is indistinguishable from a truly random value without knowledge of the mask.

Moreover, by Assumption 7.5.5, since \(\widetilde{H}_\lambda\) is an inherently sequential function family, it holds that there is no (asymptotically) more efficient way for a PPT adversary to compute \(\widetilde{h}^i_s(x)\) than to sequentially compute \(\widetilde{h}_s\) for \(t_i\) iterations. It follows that **BB-TimeLinePuzzle** is a secure time-line puzzle. \(\square\)

### 7.5.2 Concrete construction based on modular exponentiation

In this subsection we present an alternative construction quite similar in structure to the above, but based on a concrete hardness assumption. Note that the [155] time-lock puzzle construction was also based on this hardness assumption, and our time-line puzzle may be viewed as a natural "extension" of their construction.

**Assumption 7.5.8.** Let \(\text{RSA}_\lambda\) be the distribution generated as follows: sample two \(\lambda\)-bit primes \(p, q\) uniformly at random and output \(N = pq\). The family of functions \(\mathcal{H}^{\text{square}} = \{h_N : \mathbb{Z}_N \rightarrow \mathbb{Z}_N\}_{N \sim \text{RSA}_\lambda}\), where the index \(N\) is drawn from distribution \(\text{RSA}\) and \(h_N(x) = x^2 \mod N\), is inherently sequential.

**Definition 7.5.9.** **Square-TimeLinePuzzle** is a time-line puzzle defined as follows:
• Lock$_m(1^\lambda, (d_1, \ldots, d_m), (t_1, \ldots, t_m))$ takes input data $(d_1, \ldots, d_m) \in \{0, 1\}^\lambda$, samples random $\lambda$-bit primes $p, q$, sets $N = pq$, and outputs the puzzle

$$P = (x, (t_1, \ldots, t_m), N, (d_1 \oplus h_{N}^{i_1}(x), \ldots, d_m \oplus h_{N}^{i_m}(x)))$$.

• TimeStep$_m(1^\lambda, i, x', a')$ outputs $h_{a'}(x') = x'^2 \mod a'$.
• Unlock$_m(1^\lambda, x', b')$ outputs $x' \oplus b'$.

Again, it is clear that Square-TimeLinePuzzle satisfies correctness, so we proceed to prove security.

**Theorem 7.5.10.** If Assumption 7.5.8 holds, Square-TimeLinePuzzle is a secure time-line puzzle.

**Proof.** This follows from Assumption 7.5.8 in exactly the same way as Theorem 7.5.7 follows from Assumption 7.5.5, so we refer the reader to the proof of Theorem 7.5.7.  

An advantage of this construction over BB-TimeLinePuzzle is that the Lock algorithm can be much more efficient. In the case of black-box inherently sequential hash functions, we can only assume that the values $\tilde{h}_{s}^{t}(x)$ (which are XORed with the data values by the Lock algorithm) are computed by sequentially evaluating $\tilde{h}_{s}$ for $t$ iterations — that is, there is a linear dependence on $t$. However, Lock can implemented much faster with the Square-TimeLinePuzzle construction, as follows. Since $p, q$ are generated by (and therefore, available to) the Lock algorithm, the Lock algorithm can efficiently compute $\varphi(N)$. Then, $h_{N}^{t}(x)$ can be computed very efficiently by first computing $e = 2^t \mod \varphi(N)$, then computing $h_{N}^{e}(x) = x^e \mod N$. Exponentiation (say, by squaring) has only a logarithmic dependence on the security parameter.

Finally, we note that although both of the time-line puzzle constructions presented here lock $\lambda$ bits of data per puzzle (for security parameter $\lambda$), this is not at all a necessary restriction. Using encryption, it is straightforwardly possible to lock much larger amounts of data for any given parameter sizes of the time-line puzzles presented here: for example, one can encrypt the data as $\text{Enc}_k(d)$ using a secure secret-key
encryption scheme, then use the given time-line puzzle schemes to lock the key $k$ (which is much smaller than $d$) under which the data is encrypted. Such a scheme, with the additional encryption step, would be much more suitable for realistic use.

**Acknowledgements for work presented in this chapter.** We would like to thank Yehuda Lindell for an interesting discussion on the nature of fairness in multi-party computation, and we are grateful to Juan Garay, Björn Tackmann, and Vassilis Zikas for an illuminating discussion about measures of partial fairness in MPC. We thank Silvio Micali and Ron Rivest for helpful comments about the data-sharing model.
Chapter 8

Cryptographically Blinded Games: Achieving Coarse Correlated Equilibria via Cryptography

This chapter presents results from “Cryptographically Blinded Games: Leveraging Players’ Limitations for Equilibria and Profit” [102], a joint work with Pavel Hubáček that appeared in the proceedings of the 15th ACM Conference on Economics and Computation (EC), 2014.

8.1 Introduction

Nash equilibrium [144] and correlated equilibrium [8] are important solution concepts that have been extensively studied in both traditional and computational game-theoretic contexts. Coarse correlated equilibrium [143] is a closely related concept that was proposed as a generalization of correlated equilibrium, which can be more powerful in some settings such as potential games.

In this work we construct protocols for mutually distrusting players to implement any coarse correlated equilibrium (and therefore any correlated equilibrium) of a strategic game without trusted mediation, via cryptographic cheap talk protocols. Our approach draws upon cryptography in two ways: first, we introduce an interme-
Correlated equilibrium. Suppose a mediator samples an action profile \( a \) from a known distribution \( \alpha \), and gives as "advice" to each player \( i \) his action \( a_i \) in \( a \). The distribution \( \alpha \) is a correlated equilibrium if, having seen his advice, and believing that all other players will follow their advice, no player has incentive to unilaterally deviate from the advice profile. [8] showed that correlated equilibria can achieve higher expected payoffs than Nash equilibria.

Coarse correlated equilibrium. Coarse correlated equilibria are a generalization of correlated equilibria which invokes a notion of commitment. In the mediated scenario described above, \( \alpha \) is a coarse correlated equilibrium if no player has incentive not to "promise" or "commit" in advance — before seeing his advice \( a_i \) — to play according to the advice, as long as he believes that all other players will commit to do the same. Note that if a player does not commit, then he will not see the advice at all, and must therefore play an independent strategy: this is in contrast to correlated equilibria, where deviations may depend on the received advice.

[142] showed that there is a class of potential games in which the Nash equilibrium payoffs can be improved upon by coarse correlated equilibria but not by correlated equilibria (e.g., the Cournot duopoly and public good provision games).

Example 8.1.1. Let us give a brief example to illustrate the gap between the two types of equilibria. Suppose Alice plays a game \( \Gamma \) where she has a "safe strategy" for which her payoff is always zero. Let \( \alpha \) be a distribution over action profiles of \( \Gamma \), and suppose Alice's expected payoff from \( \alpha \) is very high, say, a million dollars — however, some action profiles from \( \alpha \) will give her negative payoff. Now, when Alice receives her advice from the mediator, she might be able to deduce that her payoff in the advised action profile will be negative. If this is the case, she will choose to deviate to her
safe strategy, so \( \alpha \) is not a correlated equilibrium. However, \( \alpha \) may still be a coarse correlated equilibrium if Alice can commit before seeing her advice; and importantly, \( \alpha \) may be very desirable from Alice's (risk-neutral) point of view, since expected payoff is high.

### 8.1.1 Our results

In this work we address the following question:

> How can the players of a strategic game implement any coarse correlated equilibrium via (cryptographic) pre-play communication without trusting each other or a mediator?

In the computational setting, we give an implementation for general strategic games, in the form of an extended game comprising a cryptographic protocol in the pre-play phase, which securely samples an action profile for a "cryptographically blinded" version of the original game, followed by play in the original game. The blinded game's action space consists of encryptions of the original game's actions.

Our implementation has the strong property that any computational coarse correlated equilibrium of the original game corresponds to a payoff-equivalent Nash equilibrium of the extended game. Furthermore, it achieves strategic equivalence to the original game, in that every computational Nash equilibrium of the extended game corresponds to a computational coarse correlated equilibrium of the original game. Pre-play communication is via broadcast, as is standard in the cheap talk literature.

In the information-theoretic setting, we give an implementation for strategic games with four or more players, using a similar format of a cryptographically blinded pre-play phase followed by (simultaneous) play in the original game, given private pairwise communication channels between players. As in the computational setting, we achieve strategic equivalence. Both the restriction to four or more players and the need for a stronger communication model than broadcast are unavoidable, as shown by impossibility results of [18, 9] which will be discussed in more detail in the next section.
None of our constructions require trusted mediation. After the pre-play phase is complete, there is a single step in which the players invoke a verifiable proxy to play the original game according to their instructions. Verifiable parties were introduced in [107], and will be detailed further in Section 8.1.2. No trust need be placed in the verifiable proxy, because anyone can check whether it has acted correctly; and we stress that unlike the usual mediator for coarse correlated equilibria, the verifiable proxy does not communicate anything to the players which may affect their strategies in the game. Informally, it simply performs a “translation” of a player’s chosen strategy from one form into another.

Finally, our constructions require no physical assumptions and can be executed entirely over a distributed network. This contrasts with a number of previous works such as [122, 107] which require “physical envelopes.”

8.1.2 Relation to prior work

Cheap talk. The pre-play literature considers the general problem of implementing equilibria without mediation, as follows: given an abstract game Γ, the aim is to devise a concrete communication game Γ′ having an equilibrium that is payoff-equivalent to a desirable equilibrium in Γ, where the concrete game may have a pre-play cheap talk phase in which players engage in communication that is neither costly nor binding, and has no impact on players’ payoffs except insofar as it may influence future actions. In the literature there has been much focus on implementing correlated equilibria [18, 26, 9].

Power of commitment. It has long been recognized that the possibility to commit to strategies in advance can increase the payoffs achievable in a game, starting with the work of [180], who proposed a leader/follower structure to games where the leader moves first (and thereby “commits” to his strategy). [181] showed that transforming a strategic game into a leader/follower form allows the leader (i.e., the committer) to do at least as well as in the Nash and correlated equilibria of the strategic game. Moreover, they show that coarse correlated equilibria, with their arguably stronger
notion of commitment, can yield higher payoffs than the leader/follower transformation. More recently, [123] studied the advantage of commitment from a quantitative perspective and showed that the extremal “value of commitment” is in fact unbounded in many classes of games.

In this work, we achieve the payoffs of coarse correlated equilibria without resorting to the assumption of binding contracts: instead, we use the power of encryption to hide information that, if known to the players, could render the situation unstable. We stress that the players are given the choice, rather than forced, to hide information from themselves — and we find that it is in their rational interest to do so since coarse correlated equilibria can offer high payoffs.

Cryptographic cheap talk and computational equilibria. [66] introduced the idea of cryptographic cheap talk, in which players execute a cryptographic protocol during the pre-play phase; and they defined computational equilibria, which are solution concepts stable for computationally bounded (probabilistic polynomial time) players who are indifferent to negligible gains. Their cryptographic cheap talk protocols efficiently implement some computational correlated equilibria of two-player games. Moreover, their notion of computational equilibria suffers from empty threats (Definition 8.6.5), which cause instability for sequentially rational players in the pre-play game. This was partially addressed by a new solution concept of [91]; however, [101] subsequently showed that in general, correlated equilibria cannot be achieved without empty threats by (cryptographic) cheap talk.

Our results in the computational setting use the equilibrium definitions of [66]; however, in our “cryptographically blinded” games, empty threats cannot occur. By converting games into blinded games, our constructions implement all coarse correlated equilibria without empty threats: this comes at the cost of a single mediated “translation” step using a third party, discussed in the next paragraph. We consider this step to be a “necessary” and mild requirement given that the impossibility result of [101] renders some additional assumption necessary to achieve all (coarse) correlated equilibria without empty threats.
Removing trusted mediation. Removing the need to trust a mediator in the implementation of equilibria and mechanisms has long been a subject of interest in game theory and cryptography. The notion of verifiable mediation was introduced by [107], who highlighted the difference between the usual concept of a trusted mediator, and the weaker concept of a verifiable mediator who performs actions in a publicly verifiable way and without possessing any information that should be kept secret. Recent applications of verifiable mediation include the strong correlated equilibrium implementation of [107], and the rational secret sharing scheme of [137].

In this paper, we introduce the new notion of a verifiable proxy. As in verifiable mediation, the actions of a verifiable proxy are publicly verifiable. However, our notion is incomparable to [107]'s verifiable mediation, because:

- a verifiable proxy for a strategic game does not give the players any information that affects their strategic choices in the game; and
- a verifiable proxy may possess information that should be kept secret.

More discussion about the merits of these definitions is given in Section 8.5.2.

As a simple illustration, consider a sealed-bid auction: much more trust is placed in a mediator who collects all the players' bids and just announces the winner, than in a mediator who collects the bids, opens them publicly, and allows everyone to compute the outcome themselves.

In our setting, the verifiable proxy performs a single "translation" step on behalf of the players, at the end of the pre-play phase, in which it takes strategies submitted by the players and "translates" them into a different format. In particular, the proxy acts independently and identically with respect to each player, and therefore is not implementing the correlation aspect.

Strategic equivalence property. An important concern in implementation theory is the strategic equivalence of an implementation to the underlying game: it is desirable that implementations have the "same" equilibria as the underlying game, and in particular do not introduce new ones. This was first considered by the full implementation concept of [132], and extended by subsequent works such as [107] who
proposed a stronger notion of *perfect implementation* for certain games. Although this literature is not directly applicable to the present work (as our results lie in the pre-play realm), we extend these ideas and find that the cheap talk extensions of our cryptographically blinded games achieve "best possible" strategic equivalence in that their Nash equilibria correspond exactly to the coarse correlated equilibria of their underlying games. This strategic equivalence notion is "best possible" in the sense that in the pre-play setting, the possibility of arbitrary communication in the pre-play phase inherently introduces the possibility of additional equilibria compared to the simpler one-shot game. Interestingly, Alwen et al. [41] showed that a very strong notion of strategic equivalence can be achieved if communication in the pre-play is restricted in such a way that players cannot communicate directly with each other, but only through a mediator who may "censor" some of the communication.

**Computationally unbounded setting.** To our knowledge, existing work in applying cryptographic tools to game theory has focused overwhelmingly on the setting of computationally bounded players and computational equilibria. In contrast, we consider the computationally unbounded setting too. Our result for the computational setting is stronger and more efficient than our information-theoretic solution: in particular, the computational result holds for games with any number of players, and requires only a broadcast channel for communication between players.

In the computationally unbounded setting it was proven by [18] that correlated equilibria cannot be achieved by cheap talk between fewer than four players, and indeed, this fits neatly with a more general result of [25, 51] in the context of secure protocols. Accordingly, our information-theoretic results only apply for games of four or more players; however, improving on the protocols of [18], we achieve not only correlated equilibria but coarse correlated equilibria for all games of this type.

Furthermore, in the computationally unbounded setting it has been proven [9] that communication by broadcast alone is *insufficient* to achieve (non-trivial) correlated equilibria by cheap talk, so our result is of interest notwithstanding its stronger requirement of private communication channels between players. Indeed, the private-
channels model has been extensively studied in both distributed computing (e.g., [75, 114]) and multi-party computation (e.g., [25, 51]) as an interesting strengthening of the communication model that allows for much stronger and/or more efficient protocols than the broadcast model. We therefore consider it natural and compelling to apply this model in the game-theoretical setting.

**Organization of the rest of the chapter.** In Sections 8.2 and 8.3 we provide game-theoretical and cryptographic background. In Section 8.5 we introduce cryptographically blinded games. These are the essential building block for the cheap talk protocols detailed in Section 8.6 that implement all coarse correlated equilibria of general strategic games. At the end of Section 8.6 we discuss the efficiency of our protocols.

### 8.2 Game-theoretic background

**Definition 8.2.1** (Finite strategic game). A finite strategic game \( \Gamma = (N, (A_i), (u_i)) \) is defined by a finite set \( N \) of players, and for each player \( i \in N \), a non-empty set of possible actions \( A_i \) and a utility function \( u_i : \times_{j \in N} A_j \rightarrow \mathbb{R} \).

**Remark 8.2.2.** The convention of overloading \( N \) to denote both the number of players in a game and the set of players (which can be thought to be \( \{1, \ldots, N\} \)) is standard notation in game theory. (See, e.g., [1]) In preceding chapters, we have typically denoted a set of parties or players by \([N]\) (or \([n]\)), distinguishing it from the number of players \( N \in \mathbb{N} \), as is common in the cryptography literature. Within this chapter, since we rely heavily on standard game-theoretic definitions and want to maintain compatibility with the game theory literature given the game-theoretic nature of our results, we opt for the convention of overloading \( N \) to mean both the number and set of players.

We refer to an action profile \( a = (a_j)_{j \in N} \) of a game as an outcome, and denote by \( A \) the set of outcomes \( \times_{j \in N} A_j \). For a given outcome \( a \), we write \( a_{-i} \) to denote \( (a_j)_{j \in N, j \neq i} \), that is, the profile of actions of all players other than \( i \); and we use \((a'_i, a_{-i})\)
to denote the action profile where player \( i \)'s action is \( a'_i \) and all other players' actions are as in \( a \).

### 8.2.1 Equilibrium concepts

**Definition 8.2.3** (Nash equilibrium). A Nash equilibrium of strategic game \( \Gamma = \langle N, (A_i), (u_i) \rangle \) is a product distribution \( \alpha \in \times_{j \in N} \Delta(A_j) \) such that for every player \( i \in N \) and for all \( a^*_i \in A_i \)

\[
\mathbb{E}_{a \sim \alpha} [u_i(a)] \geq \mathbb{E}_{a \sim \alpha} [u_i(a^*_i, a_{-i})].
\]

**Definition 8.2.4** (Correlated equilibrium). A correlated equilibrium of strategic game \( \Gamma = \langle N, (A_i), (u_i) \rangle \) is a probability distribution \( \alpha \in \Delta(\times_{j \in N} A_j) \) such that for every player \( i \in N \), and for all \( b_i, a^*_i \in A_i \) satisfying \( \text{Pr}_{a \sim \alpha}[a_i = b_i] > 0 \),

\[
\mathbb{E}_{a \sim \alpha} [u_i(a)|a_i = b_i] \geq \mathbb{E}_{a \sim \alpha} [u_i(a^*_i, a_{-i})|a_i = b_i].
\]

**Definition 8.2.5** (Coarse correlated equilibrium). A coarse correlated equilibrium of strategic game \( \Gamma = \langle N, (A_i), (u_i) \rangle \) is a probability distribution \( \alpha \in \Delta(\times_{j \in N} A_j) \) such that for every player \( i \in N \) and for all \( a^*_i \in A_i \)

\[
\mathbb{E}_{a \sim \alpha} [u_i(a)] \geq \mathbb{E}_{a \sim \alpha} [u_i(a^*_i, a_{-i})].
\]

The model of coarse correlated equilibrium allows the players either to "commit in advance" to play according to the mediator's advice (no matter what it turns out to be), or to play an independent strategy without learning the advice. A probability distribution is a coarse correlated equilibrium if no player has an incentive to not commit to play according to the mediator's advice.

Because of linearity of expectation, it is sufficient for these equilibrium definitions to consider only deviations to pure strategies. Note that any Nash equilibrium is a correlated equilibrium, and any correlated equilibrium is a coarse correlated equilibrium.
8.2.2 Computational equilibrium concepts

The following definitions of computational equilibria extend those introduced by [66]. In the computational setting a strategic game induces a family of games parametrized by the security parameter $\lambda$, i.e., $\Gamma = \{(N, (A_{i}^{(\lambda)}), (u_{i}^{(\lambda)}))\}_{\lambda \in \mathbb{N}}$. Hence, the corresponding solution concepts are ensembles of probability distributions, and the security parameter captures the intuition that players are limited to efficiently computable (PPT) strategies and indifferent to gains negligible in $\lambda$.

**Definition 8.2.6** (Computational Nash equilibrium). A computational Nash equilibrium of computational strategic game $\Gamma = \{(N, (A_{i}^{(\lambda)}), (u_{i}^{(\lambda)}))\}_{\lambda \in \mathbb{N}}$ is a PPT-samplable ensemble of product distributions $\alpha = \{\alpha^{(\lambda)} = \times_{j \in N} \alpha_{j}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ on $\{\times_{j \in N} A_{j}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ such that for all players $i \in N$ and every PPT-samplable ensemble $\hat{\alpha}_{i} = \{\hat{\alpha}_{i}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ on $\{A_{i}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$, there exists a negligible $\varepsilon$ such that for all $\lambda \in \mathbb{N}$ it holds that

$$\mathbb{E}_{a \leftarrow \alpha^{(\lambda)}} [u_{i}^{(\lambda)}(a)] \geq \mathbb{E}_{a \leftarrow \alpha^{(\lambda)}, \hat{a}_{i} \leftarrow \hat{\alpha}_{i}^{(\lambda)}} [u_{i}^{(\lambda)}(\hat{a}_{i}, a_{-i})] - \varepsilon(\lambda).$$

**Definition 8.2.7** (Computational correlated equilibrium). A computational correlated equilibrium of computational strategic game $\Gamma = \{(N, (A_{i}^{(\lambda)}), (u_{i}^{(\lambda)}))\}_{\lambda \in \mathbb{N}}$ is a PPT-samplable probability ensemble $\alpha = \{\alpha^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ on $\{\times_{j \in N} A_{j}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ such that for all players $i \in N$ and every PPT-samplable ensemble $\hat{\alpha}_{i} = \{\hat{\alpha}_{i}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ on $\{A_{i}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ there exists a negligible $\varepsilon$ such that for all $\lambda \in \mathbb{N}$ it holds that

$$\mathbb{E}_{a \leftarrow \alpha^{(\lambda)}} [u_{i}^{(\lambda)}(a)] \geq \mathbb{E}_{a \leftarrow \alpha^{(\lambda)}, \hat{a}_{i} \leftarrow \hat{\alpha}_{i}^{(\lambda)}} [u_{i}^{(\lambda)}(\hat{a}_{i}, a_{-i})] - \varepsilon(\lambda).$$

**Definition 8.2.8** (Computational coarse correlated equilibrium). A computational coarse correlated equilibrium of computational strategic game $\Gamma = \{(N, (A_{i}^{(\lambda)}), (u_{i}^{(\lambda)}))\}_{\lambda \in \mathbb{N}}$ is a PPT-samplable probability ensemble $\alpha = \{\alpha^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ on $\{\times_{j \in N} A_{j}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ such that for all players $i \in N$ and every PPT-samplable ensemble $\hat{\alpha}_{i} = \{\hat{\alpha}_{i}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ on $\{A_{i}^{(\lambda)}\}_{\lambda \in \mathbb{N}}$,
there exists a negligible $\varepsilon$ such that for all $\lambda \in \mathbb{N}$ it holds that

$$\mathbb{E}_{a \leftarrow \alpha^{(\lambda)}} [u_i^{(\lambda)}(a)] \geq \mathbb{E}_{a \leftarrow \alpha^{(\lambda)}, \hat{a}_i \leftarrow \hat{a}_i^{(\lambda)}} [u_i^{(\lambda)}(\hat{a}_i, a_{-i})] - \varepsilon(\lambda).$$

Note that in the above definition of computational coarse correlated equilibrium the output of $\hat{a}_i^{(\lambda)}$ is independent of $a_i$, unlike in the definition of computational correlated equilibrium.

**Remark 8.2.9.** In later sections we apply the above computational solution concepts in a straightforward way to classical strategic games. For a finite strategic game $\Gamma = \langle N, (A_i), (u_i) \rangle$ we consider the computational version $\{\Gamma^{(\lambda)}\}_{\lambda \in \mathbb{N}}$, where $\Gamma^{(\lambda)} = \Gamma$ for all $\lambda \in \mathbb{N}$. The action space and the utility function do not change with the security parameter in this computational version of $\Gamma$; however, the players are limited to efficient (PPT) strategies.

**Remark 8.2.10.** In the classical setting, it is implicit that the players of a game have oracle access to the utility functions $u_i$, that is, players can query $u_i$ on any action profile in constant time.\textsuperscript{1} Our results apply to all strategic games in the classical setting: hence the requirement that the security parameter be polynomial in the size of the game (i.e., we ensure that players are able to perform the standard task of reading the payoff matrix). With computationally bounded players, however, it seems very natural to consider the case in which computing $u_i$ takes more time. To our knowledge, this difference has been recognized (e.g., [66]) but not much analyzed in the literature; however, it is an important underlying idea of the present work.

### 8.2.3 Extensive games

Here we recall the standard definition of extensive games.

**Definition 8.2.11 (Extensive game).** An extensive game $\Gamma = \langle N, H, P, A, I, (u_i) \rangle$ is defined by:

\textsuperscript{1}Other parameters of the original game, such as the correlated equilibrium distribution, are also assumed to be computable in constant time.
• a finite set $N$ of players,
• a set $H$ of all possible history sequences (with the subset of all terminal histories denoted by $Z$),
• a player function $P : H \setminus Z \rightarrow N$ that assigns a player to every non-terminal history,
• a function $A$ that assigns to every non-terminal history $h \in H \setminus Z$ a finite set of actions $A(h) = \{ a : (h, a) \in H \}$ available to player $P(h)$ at $h$,
• for each player $i \in N$, a partition $I_i$ of $\{ h \in H : P(h) = i \}$ such that $A(h) = A(h')$ whenever $h$ and $h'$ are in the same $I_i$, and
• for each player $i \in N$, a utility function $u_i : Z \rightarrow \mathbb{R}$.

If the partition $I_i$ is trivial and each $I_i \in I_i$ contains a single history for every player $i$ then we say that the extensive game is with perfect information (i.e., every player is perfectly informed of all actions taken by every other player). A strategy profile $\sigma$ of an extensive game $\Gamma$ with perfect information specifies the actions of every player at every history, i.e., for every $h \in H$ it specifies a probability distribution on $A(h)$ for player $i = P(h)$.

The solution concept relevant to this work in the context of extensive games with perfect information is Nash equilibrium.

**Definition 8.2.12** (Nash equilibrium of extensive game). Let $\Gamma = (N, H, P, A, (u_i))$ be an extensive game with perfect information. We say that strategy profile $\sigma$ is a Nash equilibrium of $\Gamma$ if for every player $i \in N$ and for every strategy $\sigma'_i$ of player $i$:

$$\mathbb{E}[u_i(\sigma)] \geq \mathbb{E}[u_i(\sigma'_i, \sigma_{-i})],$$

where the expectations are taken over terminal histories sampled from the corresponding strategy profile.

**Definition 8.2.13** (Computational Nash equilibrium of extensive game). A computational Nash equilibrium of extensive game $\Gamma = (N, H, P, A, (u_i))$ is a PPT-samplable family of strategy profiles $\{\sigma^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ for $\Gamma$ if for every player $i \in N$ and for every
PPT-samplable strategy $\sigma_i^*$ of player $i$ it holds for all $\lambda$ that

$$\mathbb{E}[u_i(\sigma_i^{(\lambda)})] \geq \mathbb{E}[u_i(\sigma_i^*, \sigma_{-i}^{(\lambda)})] - \varepsilon(\lambda),$$

where the expectations are taken over terminal histories sampled from the corresponding strategy profile, and $\varepsilon$ is a negligible function.

In extensive games with imperfect information, the players are not informed about all the actions taken by their opponents. A profile of behavioral strategies specifies a probability distributions on actions available to every player $i \in N$ at every information set $I_i \in \mathcal{I}_i$. The solution concept of Nash equilibrium in behavioral strategies is defined similarly to Definition 8.2.12.

When reasoning about Nash equilibrium in games with imperfect information we need to take into account also the beliefs of players about the past play at any information set. This gives rise to the notion of assessment.

**Definition 8.2.14 (Assessment).** An assessment in an extensive game is a pair $(\beta, \mu)$, where $\beta$ is a profile of behavioral strategies and $\mu$ is a function that assigns to every information set a probability distribution on the histories in the information set.

The following solution concept aims to circumvent the instability of Nash equilibrium of extensive games with imperfect information.

**Definition 8.2.15 (Sequential equilibrium).** Let $\Gamma = \langle N, H, P, A, \mathcal{I}, (u_i) \rangle$ be an extensive game. The assessment $(\beta, \mu)$ is a sequential equilibrium if it is

1. sequentially rational - for every $i \in N$, for every information set $I_i \in \mathcal{I}_i$, and every $\beta_i'$

$$\mathbb{E}[u_i(\beta, \mu) | I_i] \geq \mathbb{E}[u_i((\beta_{-i}, \beta_i'), \mu) | I_i].$$

2. consistent - there exists a sequence $\{((\beta^{(n)}, \mu^{(n)}))_{n=1}\}$ of assessments that converges to $(\beta, \mu)$, $\beta^{(n)}$ is completely mixed for all $n \in N$, and $\mu^{(n)}$ is derived from $\beta^{(n)}$ by Bayes' rule.
8.3 Cryptographic background

8.3.1 Non-malleable encryption

Non-malleable encryption was introduced by [67] in the computational setting, and extended to the information-theoretic setting by [94]. Informally, non-malleability requires that given a ciphertext $c$, an adversary (who does not know the secret key or the message encrypted by $c$) cannot generate a different ciphertext $c'$ such that the respective messages are related by some known relation $R$.

We begin with the simpler information-theoretic definition. Note that [94] also give a construction of perfectly non-malleable secret-key encryption.

**Definition 8.3.1** (Perfect non-malleability). A secret-key encryption scheme $SKE = (SGen, SEnc, SDec)$ is perfectly non-malleable if for all $c, c', c'' \in \mathcal{C}$ such that $c' \neq c \neq c''$ and all relations $R : \mathcal{M} \times \mathcal{M} \to \{0, 1\}$,

$$\Pr_{sk \leftarrow SGen} [R(SDec(c), SDec(c')) = 1] = \Pr_{sk \leftarrow SGen} [R(SDec(c), SDec(c'')) = 1].$$

Perfect non-malleability implies perfect security (but not vice versa).

The computational definition of non-malleability is more involved. It formalizes the same idea, that an attacker must be unable (with more than negligible advantage) to modify ciphertexts such that the new decryption satisfies a known relation with the original decryption. The definition of non-malleability for (public-key) encryption schemes is based on the following two security experiments.

---

**The first non-malleability experiment $PubK_{AE}^{NM}(\lambda)$**

1. The challenger generates a key pair $(pk, sk) \leftarrow Gen(1^\lambda)$ and sends $(1^\lambda, pk)$ to $A$.
2. $A$ has oracle access to $Dec(sk, \cdot)$, and outputs (a description of) an efficiently sampleable distribution $M$ on the message space $\mathcal{M}$ (which must give non-zero probability only to strings of a given length).
3. The challenger samples a message $m \leftarrow M$, and sends $c \leftarrow Enc(pk, m)$ to $A$. 

306
4. A has oracle access to \( \text{Dec}^{(c)}(sk, \cdot) \), where \( \text{Dec}^{(c)} \) is defined identically to \( \text{Dec} \) except that on input \( c \) it returns \( \bot \). A outputs a ciphertext \( c' \) and (a description of) an efficiently computable relation \( R : M \times M \to \{0, 1\} \).

5. The output of the experiment is 1 if \( c' \neq c \) and \( R(m, \text{Dec}(sk, c')) \) is true, and 0 otherwise.

The second non-malleability experiment \( \text{PubK}^{\text{NM}, \$}_{A,E}(\lambda) \)

Identical to \( \text{PubK}^{\text{NM}}_{A,E}(\lambda) \), except that item 3 is replaced by:

3'. The challenger samples independent messages \( m, \tilde{m} \leftarrow M \), and sends \( c \leftarrow \text{Enc}(pk, \tilde{m}) \) to A.

**Definition 8.3.2** (Non-malleable public-key encryption). A public-key encryption scheme \( E = (\text{Gen}, \text{Enc}, \text{Dec}) \) is (computationally) non-malleable against chosen-ciphertext attacks if for all PPT adversaries \( A \) there exists a negligible function \( \varepsilon \) such that

\[
\left| \Pr[\text{PubK}^{\text{NM}}_{A,E}(\lambda) = 1] - \Pr[\text{PubK}^{\text{NM}, \$}_{A,E}(\lambda) = 1] \right| \leq \varepsilon(\lambda).
\]

In our setting\(^2\), CCA security implies computational non-malleability. For the proof, we refer the reader to [22].

**8.4 Verifiable Decryption**

Decryption can be done verifiably in any encryption scheme with recoverable randomness, a common property that is elaborated below.

The following is the standard procedure for a prover to convince a verifier that he has correctly decrypted a ciphertext in such an encryption scheme. Upon decrypting,

\(^2\)When considering security notions other than CCA, standard indistinguishability-based security does not imply non-malleability. In this work we only use CCA-secure schemes.
the prover obtains the un-encrypted action and the randomness that was used during encryption, and presents the verifier with these two items. Then the verifier can run the encryption algorithm for herself, and check that the resulting ciphertexts are the same as the ones that they submitted for decryption. By the security of the encryption scheme, it would be (computationally) infeasible for the prover to come up with (decryption, randomness) pairs that pass this check, except by running the decryption algorithm with the correct secret key. Hence, the verifier may be assured that the prover has decrypted correctly.

Note that this verifiable decryption procedure requires that the encryption scheme, in addition to being secure,\(^3\) has the following property, which is very common among existing schemes:

- **Recoverable randomness.** By running the decryption algorithm on a ciphertext \(c = \text{Enc}(m)\) with a correct secret key, the decryptor must be able to recover the randomness used for encryption. More precisely, for any given keypair \((pk, sk)\), we require that a decryptor possessing a correct secret key can efficiently compute some randomness \(r\) that, when inputted along with the correct message to the encryption algorithm, outputs the ciphertext in question, i.e., \(\text{Enc}(pk, m; r) = c\); and moreover, the decryptor cannot (with non-negligible probability) compute a randomness that, when inputted along with an incorrect message \(m'\) to the encryption algorithm, outputs the ciphertext in question, i.e., it is infeasible to find \(r'\) such that \(\text{Enc}(pk, m'; r') = c\).

In this chapter, we always refer to encryption schemes that have the property of recoverable randomness (sometimes, leaving this property implicit).

### 8.5 Cryptographically blinded games

Now we define “cryptographically blinded” games \(\Gamma'\) whose actions are encryptions of the actions of an underlying game \(\Gamma\). Payoffs from corresponding action profiles of \(\Gamma\)

\(^3\)CPA security suffices.
and Π' are the same. These blinded games will be an essential tool for our pre-play protocols, which will be detailed in Section 8.6.

The following supporting definition formalizes the intuitive notion that two strategic games are equivalent up to renaming of actions or deletion of redundant actions.

**Definition 8.5.1.** For any strategic game Π = (N, (A_i), (u_i)), a strategic game Π' = (N, (A'_i), (u'_i)) is said to be super-equivalent to Π if there exist surjective renaming functions ρ_i : A'_i → A_i such that for all i ∈ N, for all a'_i ∈ A'_i, a'_N ∈ A'_N, it holds that u'_i(a'_1, ..., a'_N) = u_i(ρ_1(a'_1), ..., ρ_N(a'_N)). In this case, we write Π' ≥_ρ Π.

**Notation.** For a renaming function ρ, let ρ⁻¹_i : A_i → Ψ(A'_i) be defined by ρ⁻¹_i(a_i) = {a'_i | ρ(a'_i) = a_i}. To simplify notation, we define ρ : A_1 × ... × A_N → A'_1 × ... × A'_N to be ρ(a_1, ..., a_N) = (ρ_1(a_1), ..., ρ_N(a_N)), and let ρ⁻¹ be defined similarly. For a distribution γ' on action profiles of Π', ρ(γ') denotes the distribution on action profiles of Π that corresponds to sampling a' ∈ A' according to γ' and outputting ρ(a').

**Lemma 8.5.2.** Let Π be a strategic game. Then for any Π' with Π' ≥_ρ Π it holds that: (1) for any coarse correlated equilibrium α of Π, there exists a coarse correlated equilibrium α' of Π' such that ρ(α') = α; and (2) for any coarse correlated equilibrium α' of Π', ρ(α') is a coarse correlated equilibrium of Π.

**Proof.** To show item (1), consider the distribution α' on action profiles of Π' obtained by sampling an action profile a from α and outputting a random a' ∈ ρ⁻¹(a). Note that ρ(α') = α by construction. We need to show that for all i ∈ N and all a'_i ∈ A'_i,

\[ \mathbb{E}_{a' \sim \alpha'} [u'_i(a')] \geq \mathbb{E}_{a' \sim \alpha'} [u'_i(a'_1, a'_{-i})]. \]

The above can be rewritten, due to the construction of α' and definition of Π', as \( \mathbb{E}_{a \sim ρ(α')} [u_i(a)] \geq \mathbb{E}_{a \sim ρ(α')} [u_i(ρ_i(a'_1), a_{-i})] \). This holds for every \( ρ_i(a'_i) \in A_i \) since \( α = ρ(α') \) is a coarse correlated equilibrium of Π. Item (2) follows similarly, since ρ(α') is a distribution on action profiles of Π and α' is a coarse correlated equilibrium. □

The interesting case of the seemingly straightforward definition of super-equivalence arises when the renaming function ρ is not invertible by the players.
We now define cryptographically blinded games. Let \( \Gamma = \langle N, (A_i), (u_i) \rangle \) be a strategic game, where players have oracle access to the utility functions \( u_i \).

**Definition 8.5.3** (Secret-key blinded game). Let \( \text{SKE} = (\text{SGen}, \text{SEnc}, \text{SDec}) \) be a secret-key encryption scheme, and let \( \Gamma = \langle N, (A_i), (u_i) \rangle \) be a strategic game. Define the blinded game \( \Gamma^{\text{SKE}} = \langle N, (A'_i), (u'_i) \rangle \) of \( \Gamma \) to be the game such that \( sk \leftarrow \text{SGen}() \) is generated and

- for each player \( i \in N \) the action space is \( A'_i = A_i \cup \{ \text{SEnc}(sk, a_i) | a_i \in A_i \} \)
- for each player \( i \in N \) the utility for all \( a' \in \times_{j \in N} A'_j \) is \( u'_i(a') = u_i(a) \), where for all \( j \in N \)
  
  \[
  a_j = \begin{cases} 
  a_j' & \text{if } a_j' \in A_j, \\
  \text{SDec}(sk, a_j') & \text{otherwise.}
  \end{cases}
  \]

**Definition 8.5.4** (Public-key blinded game). Let \( \text{E} = (\text{Gen}, \text{Enc}, \text{Dec}) \) be a public-key encryption scheme, and let \( \Gamma = \langle N, (A_i), (u_i) \rangle \) be a strategic game. Define the computational blinded game \( \Gamma^{\text{E}} = \{ \langle N, (A'^{\lambda}_i), (u'^{\lambda}_i) \rangle \}_{\lambda \in N} \) of \( \Gamma \) to be the computational game such that for every security parameter \( \lambda \in N \) a corresponding key pair \( (pk, sk) \leftarrow \text{Gen}(1^\lambda) \) is generated and

- for each player \( i \in N \) the action space is \( A'^{\lambda}_i = \{ \text{Enc}(pk, a_i) | a_i \in A_i \} \)
- for each player \( i \in N \) the utility for all \( a' \in \times_{i \in N} A'^{\lambda}_i \) is \( u'^{\lambda}_i(a') = u_i(\text{Dec}(sk, a')) \).

If \( \Gamma^{\text{E}} \) and \( \Gamma^{\text{SKE}} \) are blinded games of the game \( \Gamma \), then we say that \( \Gamma \) is the underlying game of \( \Gamma^{\text{E}} \) and \( \Gamma^{\text{SKE}} \).

Observe that the blinded games \( \Gamma^{\text{E}} \) and \( \Gamma^{\text{SKE}} \) are super-equivalent to the underlying game \( \Gamma \), with respect to renaming functions \( \rho = \text{Dec}_{sk} \) or \( \rho = \text{SDec}_{sk} \) (respectively).

**Remark 8.5.5.** In these contexts, players do not have knowledge of the secret key \( sk \), as is standard and necessary when employing encryption schemes. Therefore, expectations "from the point of view of the player" are taken over a secret key \( sk \leftarrow \text{SGen}() \) or \( (pk, sk) \leftarrow \text{Gen}(1^\lambda) \), where secret- or public-key encryption schemes are used, respectively.
It is assumed to be infeasible for players of a game $\Gamma$ to efficiently compute the utility functions $u'_i$ on arbitrary action profiles in $\Gamma^{SKE}$ or $\Gamma^E$, since they cannot (efficiently) decrypt ciphertexts in the corresponding encryption schemes. However, our applications require players to be able to pick actions in $A'_i$ for which they know the corresponding expected utility. In fact, if the players cannot do this, then the games become meaningless in that any distribution on $A$ is an equilibrium. In the public-key case, this property is achieved as players can simply compute the encryption of some $a_i \in A_i$ for which the utility is known. In the secret-key case, $A_i$ is contained in $A'_i$ for exactly this purpose.

**Security parameter for public-key games.** Public-key blinded games have an implicit security parameter $\lambda$ due to the underlying encryption scheme. When applying computational equilibrium concepts (which have a security parameter $\lambda'$ of their own) to such games, there must be a fixed relation between $\lambda$ and $\lambda'$ in order to have a meaningful definition of security for a computational equilibrium of a blinded game. In our setting, both parameters represent the same quantity: the computational boundedness of the players of a game. Therefore, we let $\lambda = \lambda'$ and refer to a single security parameter $\lambda$.

### 8.5.1 Correspondence of equilibria in blinded games

**Lemma 8.5.6.** Let $SKE = (SGen, SEnc, SDec)$ be a perfectly non-malleable and verifiably decryptable secret-key encryption scheme. Then for any strategic game $\Gamma$, it holds that for any coarse correlated equilibrium $\alpha$ of $\Gamma$ there exists a correlated equilibrium $\alpha'$ of $\Gamma^{SKE}$ that achieves the same utility profile as $\alpha$.

**Proof.** Let $\alpha'$ be the probability distribution on $\times_{i \in N} A'_i$ that corresponds to sampling an action profile $a = (a_1, \ldots, a_N) \in \times_{i \in N} A_i$ according to $\alpha$ and outputting an action profile $a' = (SEnc(sk, a_1), \ldots, SEnc(sk, a_N))$, where $sk$ is the secret key generated by $SGen$. Note that $\alpha'$ achieves the same utility profile as $\alpha$ by construction.

To show that such $\alpha'$ constitutes a correlated equilibrium of $\Gamma^{SKE}$, we need to verify that the conditions from Definition 8.2.4 are satisfied, i.e. for every player $i$
and for all $b'_i, a'_i \in A'_i$ it must hold that

\[
\mathbb{E}_{sk \leftarrow \text{SGen}, a' \leftarrow \alpha'} [u'_i(a') | a'_i = b'_i] \geq \mathbb{E}_{sk \leftarrow \text{SGen}, a' \leftarrow \alpha'} [u'_i(\hat{a}'_i, a'_{-i}) | a'_i = b'_i]. \tag{8.1}
\]

Since $\text{SKE}$ is perfectly secure, it follows from Definition 8.2 that for any $a'_0, a'_1 \in A'_i$,

\[
\mathbb{E}_{sk \leftarrow \text{SGen}, a' \leftarrow \alpha'} [u'_i(a') | a'_i = a'_0] = \mathbb{E}_{sk \leftarrow \text{SGen}, a' \leftarrow \alpha'} [u'_i(a) | a'_i = a'_1].
\]

Thus, for any player $i$, the expected utility from the distribution $\alpha'$ is independent of the advice $a'_i$. Moreover, since the underlying encryption scheme is perfectly non-malleable (Definition 8.3.1), no player $i$ can generate (with any advantage\(^4\)) a deviation $a'_i$ satisfying $R(a'_i, a_i)$ for any known relation $R$. It follows that we need only to consider deviations $a'_i$ that are independent of the received advice $a_i$. Therefore, equation 8.1 can be rewritten as the following: for every player $i$ and for all $\hat{a}'_i \in A'_i$ independent of $a'_i$,

\[
\mathbb{E}_{sk \leftarrow \text{SGen}, a' \leftarrow \alpha'} [u'_i(a')] \geq \mathbb{E}_{sk \leftarrow \text{SGen}, a' \leftarrow \alpha'} [u'_i(\hat{a}'_i, a'_{-i})],
\]

which holds because $\alpha'$ is by Lemma 8.5.2 a coarse correlated equilibrium of $\Gamma^{\text{SKE}}$. \(\square\)

**Lemma 8.5.7.** Let $E = (\text{Gen}, \text{Enc}, \text{Dec})$ be a CCA-secure public-key encryption scheme. Then for any strategic game $\Gamma$, it holds that for any computational coarse correlated equilibrium $\alpha$ of $\Gamma$ there exists a computational correlated equilibrium $\alpha'$ of $\Gamma^{E}$ that achieves the same utility profile as $\alpha$.

**Proof.** For each security parameter $\lambda \in \mathbb{N}$, let $(pk, sk)$ be the corresponding key pair generated by $\text{Gen}(1^{\lambda})$. Consider the following probability ensemble $\alpha' = \{\alpha'^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ on $\{\times_{j \in N} A_j^{(\lambda)}\}_{\lambda \in \mathbb{N}}$ that corresponds for each $\lambda \in \mathbb{N}$ to sampling an action profile $a = (a_1, \ldots, a_N) \in \times_{i \in N} A_i$ according to $\alpha^{(\lambda)}$ and outputting an action profile $a' = (\text{Enc}(pk, a_1), \ldots, \text{Enc}(pk, a_N))$. Note that $\alpha'$ achieves the same utility profile as $\alpha$ by construction.

\(^4\)More precisely, no player can generate such a deviation $a'_i$ with more success than by random guessing.
Assume that \( \alpha' \) is not a computational correlated equilibrium of \( \Gamma^E \) (Definition 8.2.7), i.e. there exist a player \( i \in N \), a PPT-samplable ensemble \( \hat{\alpha}' = \{\hat{\alpha}'(\lambda)\}_{\lambda \in \mathbb{N}} \) on \( \{A_i(\lambda)\}_{\lambda \in \mathbb{N}} \), and a non-negligible function \( \delta \) such that for every \( \lambda \in \mathbb{N} \)

\[
\mathbb{E}_{(pk, sk) \leftarrow \text{Gen}(1^\lambda), \ a' \leftarrow \alpha'(\lambda)} \left[ u_i'(\lambda)(a') \right] \leq \mathbb{E}_{(pk, sk) \leftarrow \text{Gen}(1^\lambda), \ a' \leftarrow \alpha'(\lambda), \hat{a}'_i \leftarrow \hat{\alpha}'(\lambda)(a'_i)} \left[ u_i'(\lambda)(\hat{a}'_i, a'_{-i}) \right] - \delta(\lambda).
\] (8.2)

We show that one can use such a deviation \( \hat{\alpha}'_i \) to construct a PPT adversary that contradicts the computational non-malleability of the encryption scheme \( E \) (Definition 8.3.2).

Let \( A \) be the adversary that for each security parameter \( \lambda \in \mathbb{N} \) behaves as follows. \( A \) receives a public key \( pk \) from the challenger and sends back \( M = \alpha(\lambda) \) as the message distribution. Upon receiving the challenge ciphertext \( c \) the adversary \( A \) samples \( c' \leftarrow \hat{\alpha}'(\lambda)(c) \) and sends \( c' \) to the challenger together with the relation

\[
R(b, \hat{b}) = \begin{cases} 
1 & \text{w.p. } \frac{1}{2} \cdot (\mathbb{E}_{a \leftarrow \alpha(\lambda)}[u_i(b, a_{-i})|a_i = b] - \mathbb{E}_{a \leftarrow \alpha(\lambda)}[u_i(a_i, a_{-i})|a_i = b] + 1), \\
0 & \text{otherwise}.
\end{cases}
\]

We can assume without loss of generality that all the utilities of all the players in \( \Gamma \) are between 0 and 1 (the corresponding linear transformation of the game matrix does not change the strategic properties of the game), hence the above expression defining the probability that \( R(b, \hat{b}) \) holds is between 0 and 1. Note that \( M \) is efficiently samplable and that the relation \( R \) is efficiently computable.

Consider the success probability of \( A \) in the experiment \( \text{PubK}^{\text{NM}}_{A,E}(\lambda) \), i.e.,

\[
\Pr[\text{PubK}^{\text{NM}}_{A,E}(\lambda) = 1] = \Pr_{(pk, sk) \leftarrow \text{Gen}(1^\lambda), \ a \leftarrow \alpha(\lambda), \hat{a} \leftarrow \hat{\alpha}(\lambda)(\text{Enc}(sk, a_i))} [\hat{a}' \neq \text{Enc}(sk, a_i) \land R(a_i, \text{Dec}(sk, \hat{a}'))] = \frac{1}{2} \left( \mathbb{E}_{(pk, sk) \leftarrow \text{Gen}(1^\lambda), \ a' \leftarrow \alpha'(\lambda), \hat{a}'_i \leftarrow \hat{\alpha}'(\lambda)(a'_i)} \left[ u_i'(\lambda)(\hat{a}'_i, a'_{-i}) \right] - \mathbb{E}_{(pk, sk) \leftarrow \text{Gen}(1^\lambda), \ a' \leftarrow \alpha'(\lambda)} \left[ u_i'(\lambda)(a') \right] + 1 \right) .
\]

313
Note that the scaling needed for relation $R$ is done by some finite factor, since the game matrix of $\Gamma$ does not depend on the security parameter $\lambda$. Therefore, it follows from equation 8.2 that this probability is larger than $\delta'(\lambda)$ for some non-negligible function $\delta'$.

On the other hand, the success probability of $A$ in the experiment $\text{PubK}^{\text{NM,$S$}}_{A,E}(\lambda)$, i.e.,

$$\Pr[\text{PubK}^{\text{NM,$S$}}_{A,E}(\lambda) = 1]$$

$$= \frac{1}{2} \left( \mathbb{E}_{(pk,sk) \leftarrow \text{Gen}(1^\lambda), a' \leftarrow \alpha'(\lambda), \tilde{a}_i' \leftarrow \alpha'(\lambda)} [u_i(\lambda)(\tilde{a}_i', a'_1)] - \mathbb{E}_{(pk,sk) \leftarrow \text{Gen}(1^\lambda), a' \leftarrow \alpha'(\lambda)} [u_i(\lambda)(a')] + 1 \right),$$

can be at most $\epsilon(\lambda)$ for some negligible function $\epsilon$. This follows from the fact that $\alpha$ is a computational coarse correlated equilibrium, and no independent deviation can yield a non-negligible improvement in expectation on the utility of any player $i$.

Putting the above two observations together we conclude that for some non-negligible $\delta^*$

$$\left| \Pr[\text{PubK}^{\text{NM,$S$}}_{A,E}(\lambda) = 1] - \Pr[\text{PubK}^{\text{NM,$S$}}_{A,E}(\lambda) = 1] \right| \geq \delta^*(\lambda),$$

a contradiction to computational non-malleability of $E$. \hfill $\square$

### 8.5.2 What can I do with an encrypted action?

We employ blinded games as a tool to achieve equilibria in the underlying game. The pre-play protocols in the next section will issue "advice" to the players as encrypted actions, that is, actions in the blinded game. Next, we discuss how an action of the blinded game can be "used" to take a corresponding action in the underlying game.

We return to the concept of verifiability of mediation, introduced in Section 8.1. Since the players do not know the secret key associated with a blinded game, they
cannot decrypt an encrypted action (and indeed, this is an essential property upon which the pre-play protocols will depend). The players therefore invoke a third party who plays the underlying game on their behalf. The third party will act in a way which can be publicly verified, so no trust need be placed in him to perform actions correctly: if he misbehaves, then the misconduct will be detected and he can be held accountable. This is in contrast to the usual idea of trusted mediation for implementation of equilibria.

The importance of reducing the trust placed in mediators has long been recognized in the literature, and the first formal definition of a verifiable but not trusted form of mediation was given in [107], which introduced the concept of verifiable mediator.

**Definition 8.5.8** (Verifiable mediator [107]). A verifiable mediator is a mediator which performs all actions in a publicly verifiable way, and does not use any information that must be kept secret.

We introduce the new concept of a verifiable proxy, which is used in our construction. This new concept is incomparable to the verifiable mediator of [107].

**Definition 8.5.9** (Verifiable proxy). A verifiable proxy is a mediator which performs all actions in a publicly verifiable way, and does not give the players any information that affects their strategic choices in the underlying strategic game.

In our setting, the (only) action that the verifiable proxy performs for the players is to translate the action from an encrypted form to the original form. It is important here that the proxy can perform decryption verifiably, as described in 8.4. Importantly, the verifiable proxy acts independently for each player: the correlation between players’ strategies is achieved by the players themselves with no external help, and the verifiable proxy acts simply as a proxy or interface so that the players may use encrypted actions to play in the underlying game.

We believe that (in contrast to general trusted mediators), verifiable proxies are a very realistic and mild requirement in many scenarios, since many games are already “set up” by some entity (e.g., the stock exchange or an online games company), which could easily set up instead a version of the game incorporating encrypted actions.
Moreover, the impossibility result of [101] shows that without any mediation, even correlated equilibria cannot in general be achieved by cheap talk: so some weak notion of mediation is necessary in order to bypass this result and give useful correlated equilibrium implementations.

**Example 8.5.10.** Concretely, we provide a toy example involving the well-known “Battle of the Sexes” game (Figure 8.1), where two friends are deciding on a joint activity, and they have opposing preferences but would rather be together than apart:

<table>
<thead>
<tr>
<th></th>
<th>Bach (B)</th>
<th>Stravinsky (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bach</td>
<td>2,5</td>
<td>0,0</td>
</tr>
<tr>
<td>Strav</td>
<td>0,0</td>
<td>5,2</td>
</tr>
</tbody>
</table>

Figure 8.1: “Battle of the Sexes” game

It is a correlated equilibrium to randomize over (B, B) and (S, S). In this scenario, the “encrypted advice” could be an order to an online ticket vendor for either a Bach or Stravinsky concert, encrypted under the public key of the vendor. The setup assumption here would be that the online vendor has published a public key and accepts encrypted orders. Since accepting orders in a variety of formats desirable to customers is in the vendor’s interest, we consider this to be a very feasible scenario.

Note that as this particular example is a correlated equilibrium, it is unnecessary to encrypt advice (e.g., since the protocol of [66] applies). However, the example serves to illustrate that verifiable translation can be a highly realistic and mild assumption.

**8.6 Our Protocols**

In this section we give cryptographic protocols (in the computational and information-theoretic settings) that achieve the utility profile of any coarse correlated equilibrium.

**8.6.1 Cryptographic cheap talk**

**Definition 8.6.1** (Cheap talk extension). For a strategic game \( \Gamma \), the cheap talk extension \( \tilde{\Gamma} \) is defined as an extensive game consisting of a pre-play phase in which
the players exchange messages, followed by the play in the original strategic game. The communication is non-binding (unlike in signaling games) in that it does not directly affect players’ utilities in the underlying game, that is, players’ utilities in the cheap talk extension depend only on actions taken in the strategic game. The cryptographic cheap talk extension is defined exactly like the cheap talk extension, except that the players exchange messages during a polynomially bounded number of rounds prior to the play in the original game $\Gamma$.

We follow the pre-play paradigm of [18], where the mediator is replaced by “cheap talk” communication prior to game play. We construct protocols to be run during pre-play, which implement any (computational) coarse correlated equilibrium of blinded games as a (computational) Nash equilibrium of the (computational) cheap talk extension.

### 8.6.2 Protocol for computationally bounded players

In this protocol, the players run a computationally secure multi-party computation to sample an action profile from any computational correlated equilibrium of the blinded game.

**Protocol 4** Implementing any computational correlated equilibrium $\alpha'$ of $\Gamma^E$

Let $E = (\text{Gen}, \text{Enc}, \text{Dec})$ be a CCA-secure public-key encryption scheme and let $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ with $pk$ known to all players. Communication is via broadcast.

1. The players run a computationally secure multi-party computation protocol (secure against $t \leq N - 1$ corruptions) to implement the function that samples an action profile $a' \leftarrow \alpha'$, and outputs to each player $i$ his action $a'_i$.
2. Every player takes $a'_i$ as its action in $\Gamma^E$.

We show that rational computationally bounded players will follow the above protocol, so they can use it to implement any computational correlated equilibrium. Then, by combining the above with our results from Section 8.5 about correspondence of coarse correlated equilibria in the underlying game and correlated equilibria in its blinded version, we obtain that the protocol can moreover be used to implement any computational coarse correlated equilibrium.
Note that it is necessary to treat the two-player case somewhat differently from the case with three or more players, because of the problem of guaranteed output delivery in the two-player case (which was described in Section 2.7). We begin by presenting the simpler Theorem 8.6.2, which states that Protocol 4 can be directly used by three or more players to implement any computational coarse correlated equilibrium. Then, we give Theorems 8.6.3 and 8.6.4 which show that by running a slightly modified version of Protocol 4, it is possible for any number of players to implement any computational coarse correlated equilibrium.

**Theorem 8.6.2.** Let \( E = (\text{Gen}, \text{Enc}, \text{Dec}) \) be a CCA-secure public-key encryption scheme, and let \( \Gamma \) be any finite strategic game with three or more players. For any computational coarse correlated equilibrium \( \alpha \) of \( \Gamma \), there exists a computational Nash equilibrium \( \tilde{\alpha} \) of the computational cheap talk extension \( \Gamma^E \) that achieves the same utility profile as \( \alpha \).

**Proof.** Let \( \alpha' \) be the computational correlated equilibrium of \( \Gamma^E \) from Lemma 8.5.7 that achieves the same utility profile as \( \alpha \). We show that using Protocol 4 in order to implement \( \alpha' \) constitutes a computational Nash equilibrium in the cryptographic cheap talk extension \( \Gamma^E \). Note that it is payoff-equivalent to \( \alpha \) by construction.

By the secrecy guarantee of the secure multi-party computation protocol, we have that no player can learn any (non-negligible amount of) information that cannot be deduced from his intended output in the first place, even if he deviates from the protocol arbitrarily. Moreover, since there are three or more players and we consider only unilateral\(^5\) deviations (as implied by the definition of Nash equilibrium), the protocol has the property of guaranteed output delivery.\(^6\) therefore, the deviation of any player \( i \) cannot prevent any other player \( j \) from receiving her correct output \( a'_j \).

We have shown that for any player, there is no deviation during the protocol phase that is profitable by more than negligible amount. Hence, we consider only the

---

\(^5\)That is, we only consider deviations from the protocol by a single (malicious) player, rather than by coalitions of multiple colluding players.

\(^6\)We remark that in fact, the slightly weaker property of fairness is sufficient: that is, the property that if any player receives his output in the protocol, then every honest player will receive her correct output too. However, in the settings we consider, the stronger property of guaranteed output delivery is known to hold, hence we refer to the latter property in order to slightly simplify the proof.
case where each player $i$ receives his correct output $a_i'$. Since $\alpha'$ is, by Lemma 8.5.7, a computational correlated equilibrium of $\Gamma^E$, no player has an incentive to deviate from the prescribed advice, and thus the players will play according to the sampled action profile $a'$. Therefore, to follow Protocol 4 is the computational Nash equilibrium $\bar{\alpha}$ of $\Gamma^E$ payoff-equivalent to $\alpha$.

Dealing with the two-player case

In the two-player case, the additional complication stems from the fact that in this setting we do not have guaranteed output delivery: hence, it is necessary to consider that a player may be incentivized to cause a protocol execution to terminate prematurely. In order to disincentivize such behavior, we introduce an additional "punishment" condition to the protocol, as follows.

**Protocol 5** Implementing any computational correlated equilibrium $\alpha'$ of $\Gamma^E$

Let $E = (\text{Gen}, \text{Enc}, \text{Dec})$ be a CCA-secure public-key encryption scheme and let $(pk, sk) \leftarrow \text{Gen}(1^\Lambda)$ with $pk$ known to all players. Communication is via broadcast.

- The players run Protocol 4 as long as no player is detected to deviate from the protocol.
- If any player $i$ is detected to deviate from the protocol, all (other) players adopt the strategies (in $\Gamma^E$) corresponding to the worst Nash equilibrium $\sigma^i$ for player $i$.

Using Protocol 5, we obtain the following theorem that applies for any number of players.

**Theorem 8.6.3.** Let $E = (\text{Gen}, \text{Enc}, \text{Dec})$ be a CCA-secure public-key encryption scheme, and let $\Gamma$ be any finite strategic game. For any computational coarse correlated equilibrium $\alpha$ of $\Gamma$ that for each player achieves at least as high utility as the worst Nash equilibrium, there exists a computational Nash equilibrium $\bar{\alpha}$ of the computational cheap talk extension $\Gamma^E$ that achieves the same utility profile as $\alpha$.

**Proof.** Let $\alpha'$ be the computational correlated equilibrium of $\Gamma^E$ from Lemma 8.5.7 that achieves the same utility profile as $\alpha$. We show that using Protocol 5 in order to implement $\alpha'$ constitutes a computational Nash equilibrium in the cryptographic
cheap talk extension $\widetilde{\Gamma}^E$. For any security parameter $\lambda$, the following events may occur during the run of the protocol:

1. a player learns its advice before the other players;
2. a player deviates from the protocol and the deviation is detected by the other players; or
3. a player deviates from the protocol and it is unnoticed.

Addressing (1): it follows from CCA security of the public-key encryption scheme $E$ (Definition 2.5.4) that each player is indifferent (up to a negligible improvement in utility) between any advice he may receive, and thus gains no advantage from learning his advice first. In particular, he has no incentive to abort the protocol and prevent others from learning their advice. Addressing (2): the expectation of any player $i$ in the default Nash equilibrium $\sigma^i$ is at most the expectation of player $i$ in $\alpha$. Addressing (3): the security of the multi-party computation protocol ensures that players can cheat without being caught with at most negligible probability. Thus, the increase in utility from any cheating strategy is at most negligible.

There is no deviation during the protocol phase profitable by more than negligible amount. Consider the case that every player $i$ received his advice $a'_i$. Since $a'_i$ is, by Lemma 8.5.7, a computational correlated equilibrium of $\widetilde{\Gamma}^E$, no player has an incentive to deviate from the prescribed advice, and the players will play according to the sampled action profile $a'$. Therefore, to follow Protocol 5 is the computational Nash equilibrium $\widetilde{\alpha}$ of $\widetilde{\Gamma}^E$ payoff-equivalent to $\alpha$.

It is possible to eliminate the condition (from Theorem 8.6.3) that the implemented coarse correlated equilibrium does at least as well as the respective worst Nash equilibrium for each player, thereby obtaining a yet more general theorem as follows.

**Theorem 8.6.4.** Let $E = (\text{Gen, Enc, Dec})$ be a CCA-secure public-key encryption scheme, and let $\Gamma$ be any finite strategic game. For any coarse correlated equilibrium $\alpha$ of $\Gamma$, there exists a computational Nash equilibrium $\widetilde{\alpha}$ of the computational cheap talk extension $\widetilde{\Gamma}^E$ that achieves the same utility profile as $\alpha$. 320
The proof of Theorem 8.6.4 makes use of another variant of Protocol 4. The details of this variant protocol are given next, in Protocol 6. A proof sketch of Theorem 8.6.4 follows.

**Protocol 6** Implementing any computational correlated equilibrium $\alpha'$ of $\Gamma^E$

Let $E = (\text{Gen}, \text{Enc}, \text{Dec})$ be a CCA-secure public-key encryption scheme and let $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ with $pk$ known to all players. Communication is via broadcast.

- The players run Protocol 4 as long as no player is detected to deviate from the protocol.
- If any player $i$ is detected to deviate from the protocol, all (other) players adopt joint min-max strategy (in $\Gamma^E$) with the worst possible outcome for player $i$.

**Proof (sketch).** The players run Protocol 6, in which — by construction — adhering to the protocol yields at least as much utility as deviation, for any given player. The full proof follows exactly the same structure as the proof of Theorem 1 of [66], and we refer the reader to their paper for details.

**Empty threats.** As discussed briefly in Section 8.6.2, implementation via Protocol 6 has the disadvantage (compared to Protocol 5) of empty threats. We first give an informal definition and discussion of empty threats, then proceed to a definition and proof sketch of the empty-threat-freeness of Protocol 5.

**Definition 8.6.5** (Empty threat (informal)). An empty threat posed by a player in an extensive game is a strategy of the threatening player at a history off the equilibrium path which is not rational from his perspective. A threatened player can demonstrate the existence of such an empty threat by taking a beneficial deviation that would make the threatening player refrain from following through with the announced threat.

A consequence of empty threats in a Nash equilibrium is that a strategy profile containing empty threats is not sequentially stable, that is, players that adapt their strategies during the game would not follow such a strategy profile. One approach to avoid empty threats is to require subgame perfect equilibrium. However, as addressed by [91], there is no obvious way to define subgame perfection in the computational
setting. Therefore, we use the computational solution concept of "empty-threat-free Nash equilibrium" [91] and argue that Protocol 5 is free of empty threats.

Now, we give the definition of empty-threat-free Nash equilibrium in extensive games, which we use to give a proof sketch of the empty-threat-freeness of our Protocol 5. The computational version of this solution concept is much more definitionally involved; we refer to [91] for full details of the computational definition.

We use the following notion of set of continuations of a strategy profile at a given history.

**Definition 8.6.6 (Continuation).** For a history \( h \in H \), a strategy \( \sigma \), and a distribution \( \tau = \tau(h) \) on \( A(h) \), let

\[
\text{Cont}(h, \sigma, \tau) = \{ \pi : (\pi \text{ differs from } \sigma \text{ only on the subgame } h) \land (\pi(h) = \tau(h)) \}.
\]

**Definition 8.6.7 (Empty threat).** Let \( \Gamma = (N, H, P, A, (u)_i) \) be an extensive game, and let \( \sigma \) be a strategy profile. Then:

- For any history \( h \in Z \), no player faces an empty threat at \( h \) with respect to \( \sigma \).
- Player \( i \) faces an empty threat at history \( h \in H \setminus Z \) with respect to \( \sigma \) if \( i = P(h) \) and there exists a distribution \( \tau = \tau(h) \) over \( A(h) \) that satisfies the following: for all \( \pi \in \text{Cont}(h, \sigma, \tau) \) and \( \pi' \in \text{Cont}(h, \sigma, \sigma) \) for which no player faces an empty threat at any \( h' \in H \) below \( h \), it holds that

\[
E[u_i(\pi)] > E[u_i(\pi')].
\]

A strategy profile \( \sigma \) is empty-threat-free on \( h \) if for all \( h' \neq \emptyset \) satisfying \( (h, h') \in H \) no player faces an empty threat at \( (h, h') \) with respect to \( \sigma \).

**Definition 8.6.8 (Empty-threat-free Nash equilibrium).** Let \( \Gamma = (N, H, P, A, (u)_i) \) be an extensive game. Strategy profile \( \sigma \) is an empty-threat-free Nash equilibrium if:

- \( \sigma \) is a Nash equilibrium of \( \Gamma \), and
- for any \( h \in H \setminus Z \), player \( P(h) \) does not face an empty threat at \( h \) with respect to \( \sigma \).
Remark 8.6.9. The above definitions readily apply to extensive games with \( n > 2 \) players, even though they were originally intended for games with two players. As noted by [91J, a potential shortcoming of applying this definition in games with more than two players is that it does not take into account collusions between players. In other words, the definition addresses only unilateral deviations, as is accepted and standard in Nash equilibrium.

Theorem 8.6.10. Let \( \tilde{\alpha} \) be the computational Nash equilibrium of \( \Gamma^E \) from Theorem 8.6.3. Then \( \tilde{\alpha} \) is an empty-threat-free computational Nash equilibrium of \( \Gamma^E \).

Proof (sketch). Recall that the computational Nash equilibrium \( \tilde{\alpha} \) is payoff-equivalent to some computational coarse correlated equilibrium \( \alpha \) of \( \Gamma \) (Theorem 8.6.3), and that \( \alpha \) achieves at least as high utility for each player \( i \) as his worst Nash equilibrium \( \sigma^i \) of \( \Gamma \) to which the players default in case any deviation of player \( i \) is detected during the protocol phase.

For every security parameter \( \lambda \), we need to show that at any history no player is facing an empty threat (see Definition 8.6.7), i.e. we need to show that there is no history \( h \) with a deviation \( \tau \) such that every empty-threat-free continuation of \( \tau \) improves over every empty-threat-free continuation of \( \tilde{\alpha} \) at \( h \) by more than negligible amount.

It follows from Lemma 8.5.7 that the expectation of any player after receiving the encrypted advice is the same as the expectation of playing \( \tilde{\alpha} \) without knowing the advice. Thus, the expectation from following the protocol is the same at any history of the cheap talk extension.

We will use the following claim that follows immediately from \( \tilde{\alpha} \) being a computational Nash equilibrium.

Claim 8.6.11. Any deviation during the protocol phase that goes unnoticed can give the player at most negligible advantage.

Since any observed deviation corresponds to a history in which players default to the Nash equilibrium \( \sigma \), no player is facing an empty threat with respect to \( \tilde{\alpha} \) at such histories, since \( \sigma \) is a Nash equilibrium.
By the definition of empty threat (Definition 8.6.7), no player is facing an empty threat at the final round where players take simultaneous actions in the strategic game, and in particular it is an empty-threat-free strategy to play according to the received advice \( a'_t \) at the terminal history. By Claim 8.6.11 any unobserved deviation in the protocol phase can yield at most negligible improvement in player’s utility, thus we get by induction that to follow \( \tilde{\alpha} \) is an empty-threat-free continuation at any history.

Finally, Claim 8.6.11 also gives that no continuation (in particular no empty-threat-free continuation) of any deviation at any history \( h \) can improve by more than negligible amount over the continuation induced at \( h \) by following \( \tilde{\alpha} \). Therefore, \( \tilde{\alpha} \) is an empty-threat-free computational Nash equilibrium of \( \Gamma^E \).

\[ \square \]

**Discussion of protocols.** Protocol 5 has certain more desirable properties than Protocol 6: in particular, Protocol 5 is free of empty threats, which ensures that Nash equilibria in the protocol are stable even when players may change strategy adaptively during protocol execution. Ultimately, notwithstanding the restriction on the class of achieved coarse correlated equilibria, we consider Theorem 8.6.3 to be the much stronger result compared to Theorem 8.6.4, for the following reasons:

- all coarse correlated equilibria that players might rationally wish to implement by cheap talk do dominate all Nash equilibria (otherwise, they could achieve a better payoff from a Nash equilibrium without the hassle of a pre-play protocol);
- unlike Protocol 6, Protocol 5 is free of empty threats; and
- the expected payoff even when the protocol is aborted and the default strategy invoked is higher in Protocol 5 than in Protocol 6.

**Strategic equivalence.** Lemma 8.6.12, below, proves the strategic equivalence of the cryptographic cheap talk extension \( \Gamma^E \) to the underlying game \( \Gamma \).

**Lemma 8.6.12.** Let \( E = (\text{Gen, Enc, Dec}) \) be a CCA-secure public-key encryption scheme, and let \( \Gamma \) be any finite strategic game. For any computational Nash equilibrium \( \tilde{\alpha} \) of the cryptographic cheap talk extension \( \Gamma^E \), there exists a computational
coarse correlated equilibrium $\alpha$ of $\Gamma$ that achieves the same utility profile as $\tilde{\alpha}$.

Proof. We show that the probability ensemble $\alpha$ induced by $\tilde{\alpha}$ on action profiles of $\Gamma$ is a computational coarse correlated equilibrium of $\Gamma$.

Assume that $\alpha$ is not a computational coarse correlated equilibrium, i.e., there exists a player $i$ that has a PPT-samplable unilateral deviation to $\alpha$ that improves his expectation for every $\lambda \in \mathbb{N}$ by $\delta(\lambda)$ for some non-negligible $\delta$. However, such deviation can be used by player $i$ also against $\tilde{\alpha}$ to gain a non-negligible improvement in his expectation in $\tilde{\Gamma}^E$, a contradiction to the fact that $\tilde{\alpha}$ is a computational Nash equilibrium of $\tilde{\Gamma}^E$.

Corollary 8.6.13. For any finite strategic game $\Gamma$, the cryptographic cheap talk extension $\tilde{\Gamma}^E$ is strategically equivalent to $\Gamma$, that is, for every Nash equilibrium $\tilde{\alpha}$ of $\tilde{\Gamma}_{SKE}$, there exists a coarse correlated equilibrium of $\Gamma$ that achieves the same utility profile as $\tilde{\alpha}$, and vice versa.

Proof. Follows immediately from Lemma 8.6.12 and Theorem 8.6.3 (or Theorem 8.6.2 for the case of three or more players).

8.6.3 Protocol for computationally unbounded players

An alternative protocol using secret-key encryption implements all coarse correlated equilibria — not just computational ones — for all strategic games with four or more players. As discussed in Section 8.1, the condition of four or more players is unavoidable. In this (more traditional) setting, the players are computationally unbounded.

Protocol 7 Implementing any correlated equilibrium $\alpha'$ of $\tilde{\Gamma}_{SKE}$

Let $SKE = (SGen, SEnc, SDec)$ be a perfectly non-malleable and verifiably decryptable secret-key encryption scheme and let $sk \leftarrow SGen$. Let each player $i$ possess a distinct share $sk_i$ of an $(N - 1)$-out-of-$N$ secret-sharing $\{sk_1, \ldots, sk_N\}$ of $sk$. Communication is via pairwise channels.

1. The players run a perfectly secure multi-party computation to implement the function that samples a profile $\alpha' \leftarrow \alpha'$, and outputs to each $i$ his action $a'_i$.
2. Every player takes $a'_i$ as its action in $\Gamma_{SKE}$.
Theorem 8.6.14. Let $\text{SKE} = (\text{SGen}, \text{SEnc}, \text{SDec})$ be a perfectly non-malleable and verifiably decryptable secret-key encryption scheme, and let $\Gamma$ be any finite strategic game with four or more players. For any coarse correlated equilibrium $\alpha$ of $\Gamma$ there exists a Nash equilibrium $\tilde{\alpha}$ of the cheap talk extension $\tilde{\Gamma}^{\text{SKE}}$ that achieves the same utility profile as $\alpha$.

Proof. Let $\alpha'$ be the correlated equilibrium of $\Gamma^{\text{SKE}}$ from Lemma 8.5.6 that achieves the same utility profile as $\alpha$. We show that to follow Protocol 7 in order to implement $\alpha'$ constitutes the Nash equilibrium $\tilde{\alpha}$ in the cryptographic cheap talk extension $\tilde{\Gamma}^{\text{SKE}}$ that achieves the same utility profile as $\alpha$.

First note that since the players are using a perfectly secure protocol with guaranteed output delivery to implement sampling from $\alpha'$, no player can prevent the others from learning their advice by a unilateral deviation during the multi-party computation phase. Moreover, even if a single player $i$ withholds its share $sk_i$ the remaining players hold $N - 1$ shares of the secret key $sk$ that are sufficient to reconstruct the secret key and sample an action profile from $\alpha'$. Hence, any unilateral deviation does not influence the distribution on actions taken by the other players. Assume that there exists a unilateral deviation for some player $i$ in $\tilde{\Gamma}^{\text{SKE}}$ that allows him to gain a higher utility than by playing according to $\tilde{\alpha}$. This contradicts $\alpha'$ being a correlated equilibrium of $\Gamma^{\text{SKE}}$, since it could be used as a unilateral profitable deviation against $\alpha'$ in $\Gamma^{\text{SKE}}$ as well. $\square$

Strategic equivalence. Lemma 8.6.15, below, proves the strategic equivalence of the cheap talk extension $\tilde{\Gamma}^{\text{SKE}}$ to the underlying game $\Gamma$.

Lemma 8.6.15. Let $\text{SKE} = (\text{SGen}, \text{SEnc}, \text{SDec})$ be a perfectly non-malleable and verifiably decryptable secret-key encryption scheme, and let $\Gamma$ be any finite strategic game with four or more players. For any Nash equilibrium $\tilde{\alpha}$ of the cheap talk extension $\tilde{\Gamma}^{\text{SKE}}$, there exists a coarse correlated equilibrium $\alpha$ of $\Gamma$ that achieves the same utility profile as $\tilde{\alpha}$.

Proof. We show that the distribution $\alpha$ induced by $\tilde{\alpha}$ on action profiles of $\Gamma$ is a coarse correlated equilibrium of $\Gamma$. Suppose $\alpha$ is not a coarse correlated equilibrium,
i.e., there exists a player \( i \) that has a deviation to \( \alpha \) which improves his expectation. However, such a deviation contradicts the fact that \( \tilde{\alpha} \) is a Nash equilibrium of \( \Gamma^{\text{SKE}} \), since it is also a profitable unilateral deviation against \( \tilde{\alpha} \) in \( \Gamma^{\text{SKE}} \).

**Corollary 8.6.16.** For any game \( \Gamma \), it holds that the cheap talk extension \( \Gamma^{\text{SKE}} \) is strategically equivalent to \( \Gamma \), that is, for every Nash equilibrium \( \tilde{\alpha} \) of \( \Gamma^{\text{SKE}} \), there exists a coarse correlated equilibrium of \( \Gamma \) that achieves the same utility profile as \( \tilde{\alpha} \), and vice versa.

*Proof.* Follows immediately from Theorem 8.6.14 and Lemma 8.6.15.

**Sequential equilibrium.** We also show that the equilibrium from Theorem 8.6.14 is a *sequential equilibrium*: informally, we show that by following the prescribed strategy, the players are making optimal decisions at all points in the game tree. Our proof relies on perfect security for multi-party computation protocols in the presence of one actively corrupted and one passively corrupted party which can be achieved only for six or more players (as shown by Fitzi, Hirt and Maurer [76]). Hence, the statement of the following theorem is less general than the statement of Theorem 8.6.14.

**Theorem 8.6.17.** Let \( \text{SKE} = (\text{SGen}, \text{SEnc}, \text{SDec}) \) be a perfectly non-malleable and verifiably decryptable secret-key encryption scheme, and let \( \Gamma \) be any finite strategic game with six or more players. For any coarse correlated equilibrium \( \alpha \) of \( \Gamma \) there exists a sequential equilibrium \((\tilde{\alpha}, \mu)\) of the cheap talk extension \( \Gamma^{\text{SKE}} \) that achieves the same utility profile as \( \alpha \).

*Proof.* We assume without loss of generality that the multi-party computation protocol has the canonical structure where at each round a single player receives a message from one of the other players (i.e., the information sets in the extensive game correspond to histories consistent with the received message). Since there is at least six players, we can assume that multi-party computation is secure in the presence of one static and one active corruption. Consider the behavioral strategy profile \( \tilde{\alpha} \).
corresponding to following Protocol 7 at each history where a player receives a message from some other player (in particular this corresponds to ignoring all received messages after termination of the multi-party computation).

First, we specify the belief system $\mu$ of players at any information set. The beliefs at any information set on the equilibrium path are derived from the behavioral strategy $\tilde{\alpha}$ by Bayes’ rule, and for any information set $I$ that lies off the equilibrium path (i.e., an information set corresponding to receiving a message out of the scope of the protocol), let $\mu(I)$ be the uniform distribution on all histories in $I$. To show that $(\tilde{\alpha}, \mu)$ is a sequential equilibrium, we must show that $(\tilde{\alpha}, \mu)$ is both sequentially rational and consistent.

Since $\tilde{\alpha}$ is a Nash equilibrium (as shown in Theorem 8.6.14), the behavioral strategy to follow $\tilde{\alpha}$ is optimal for any information set on the equilibrium path. Hence, to conclude that $(\tilde{\alpha}, \mu)$ is sequentially rational, we just need to show that $\tilde{\alpha}$ is also optimal off the equilibrium path, given the beliefs of $\mu$. Let $I$ be an information set of player $i$ at some point off the equilibrium path that corresponds to receiving a message from player $j$. Note that even if $j$ sends to $i$ its complete view of the protocol up to this point player $i$ cannot use such information to produce a profitable deviation, since such deviation would imply an adversary corrupting actively player $i$ and statically player $j$ able to break the perfect security of the multi-party computation protocol. Now consider any history off the equilibrium path after the termination of the multi-party computation, and assume that player $i$ receives the private advice of some other player. There cannot exist a profitable deviation of player $i$, since such a deviation would contradict security of the secret key encryption scheme.

To show that $(\beta, \mu)$ is consistent we use the “trembling-hand” approach. Consider the sequence of assessments $\{((\beta^{(n)}, \mu^{(n)}))\}_{n=1}^{\infty}$ where each $\beta^{(n)}$ assigns non-zero probability $e^{(n)}$ to all actions that are taken with zero probability in $\beta$, such that $e^{(n)}$ goes to zero as $n \to \infty$, and the belief system $\mu^{(n)}$ is derived from $\beta^{(n)}$ using Bayes’ rule. First note that the sequence $\{((\beta^{(n)}, \mu^{(n)}))\}_{n=1}^{\infty}$ converges to $(\beta, \mu)$. The sequence of behavioral strategy profiles $\{\beta^{(n)}\}_{n=1}^{\infty}$ converges to $\beta$ by construction. Since $\mu^{(n)}$ is derived from $\beta^{(n)}$ by the Bayes’ rule, $\mu^{(n)}$ converges to $\mu$ for every information set
on the equilibrium path. For every information set $I$ off the equilibrium path, the distribution $\mu^{(n)}(I)$ is equal to $\mu(I)$. Finally, $\beta^{(n)}$ is completely mixed for all $n$, hence $(\beta, \mu)$ is consistent.

8.6.4 Remarks on efficiency of multi-party computation

Computational setting. With recent advances in efficiency, computationally secure multi-party computation protocols are now being considered for practical use in various settings. Its first large-scale deployment was to compute market clearing prices for Danish sugar beet contracts in 2008 [40]. Subsequent advances include [105, 59]. Indeed, numerous multi-party computation implementations are available online, such as VIFF (viff.dk) [57].

In the common "pre-processing model," where pre-processing time is available before the main computation, yet faster protocols are possible: [60] achieves secure 3-party 64-bit multiplication in 0.05 ms. This could be a very reasonable model when the same $N$ players play multiple or repeated games.

We note that there has been a line of work starting with [66], on designing multi-party computation protocols specifically for sampling from correlated equilibrium distributions. However, these address the two-party setting, and have not taken into account the most recent advances in general multi-party computation techniques, so we do not consider them to be of great relevance here.

Perfect setting. In the perfect setting, known protocols are less efficient; and perfectly secure encryption is relatively inefficient due to inherently large key sizes. Nonetheless, substantial progress has been made: the best known protocol [20] achieves $O(N)$ communication complexity per multiplication, improving on previous protocols by $\Omega(N^2)$.

We consider our information-theoretic results to be of interest primarily as proofs of possibility, and a novel application of cryptographic techniques to game theory.

---

7 The circuit that the parties want to compute is usually represented as addition and multiplication gates, and the multiplication gates have been found to be the bottleneck for multi-party computation.
without computational restrictions. Certainly, for efficiency in practice and strength of results, our computational protocols are the ones of interest.

8.7 Conclusion

In this work we use standard cryptographic tools — namely, encryption schemes — to introduce the concept of blinded games: strategic games in which players take encrypted actions, and in particular have the possibility to take actions they know nothing about. Moreover, we provide cryptographic protocols that enable the players to not rely on trusted mediators in order to achieve equilibrium payoffs.

Our approach suggest new interesting uses of cryptographic methods in game theory. We show that our blinded games offer a viable and appealing alternative to solution concepts based on commitment, and a particularly promising direction for future work is to apply the paradigm of leveraging players' lack of knowledge in order to avoid commitment, in broader settings.

Acknowledgements for work presented in this chapter. We are grateful to Alessandra Scafuro for raising the question of encrypting advice, to Silvio Micali and David Parkes for very helpful general advice on draft versions, and to Jesper Buus Nielsen for detailed technical comments.
Bibliography


346


