Keep it Secret, Keep it Safe

Privacy, Security, and Robustness in an Adversarial World

by

Adam Benjamin Gelernter Sealfon

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

Signature redacted

Author ....
Department of Electrical Engineering and Computer Science
August 30, 2019

Signature redacted

Certified by.. ....
Shafi Goldwasser
RSA Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by ....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Keep it Secret, Keep it Safe

Privacy, Security, and Robustness in an Adversarial World

by

Adam Benjamin Gelernter Sealfon

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

The deployment of large-scale systems involving many individuals or devices necessitates the design of computational frameworks that are resilient to failures or malicious actors. This thesis introduces algorithms and definitions for a series of problems concerning robustness, security, and privacy in the many-party setting. We describe protocols for maintaining a stable configuration despite adversarial perturbations, for cryptographic tasks involving secure multiparty computation and anonymity-preserving authentication, and for privacy-preserving analysis of networks. The results presented span the fields of distributed algorithms, cryptography, and differential privacy.

We first model and describe a protocol for the problem of robustly preserving a stable population size in the presence of continual adversarial insertions and deletions of agents. Turning to cryptography, we explore the possibility of leveraging an infrastructure for secure multiparty computation, characterizing which networks of pairwise secure computation channels are sufficient to achieve general secure computation among other sets of parties. We next introduce a definitional framework and constructions for ring signatures that provide more fine-grained functionality, explicitly delineating whether parties can convincingly claim or repudiate authorship of a signature. Finally, we turn to differential privacy for graph-structured data. We present efficient algorithms for privately releasing approximate shortest paths and all-pairs distances of a weighted graph while preserving the privacy of the edge weights. We also present efficient node-private algorithms for computing the edge density of Erdős-Rényi and concentrated-degree graphs.

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

Acknowledgments

Firstly, I am grateful to my advisor, Shafi Goldwasser, for her tireless guidance throughout my PhD. I've learned so much from her, and deeply appreciate her dedication, encouragement, and mentorship.

I'd like to thank the other members of my thesis committee, Ron Rivest and Jon Ullman. I am fortunate to have worked with each of them and to have learned from them in classes and through research collaborations.

I'd also like to thank Adi Akavia, Yael Tauman Kalai, David Peleg, Alon Rosen, Salil Vadhan, and Vinod Vaikuntanathan for mentoring me either at MIT or during research visits to their institutions. I'd especially like to thank David and Salil for advising me as an undergraduate.

Thanks to all of my other collaborators, including Ranjit Kumaresan, Rafi Ostrovsky, Sunoo Park, Srini Raghuraman, Ron D. Rothblum, Alessandra Scafuro, and Katerina Sotiraki, for the wonderful experience of working with them and for teaching me so much. This thesis is based on papers coauthored with many of them.

Thanks to my labmates in MIT's Theory of Computation group for being such great friends and colleagues over the years.

Finally, I'd like to thank my family for all of their support and love throughout my PhD, before, and beyond. This thesis is dedicated to them.

Contents

List of Figures

Chapter 1

Introduction

The computing landscape over the past decades has become increasingly globalized. The online ecosystem has enabled the execution of joint computations performed in tandem by many participants at physically dispersed locations, and has also facilitated the collection and analysis of enormous datasets describing many millions of individuals. At the same time, this increasingly decentralized paradigm for computing at scale raises novel challenges for security, privacy, and robust accuracy, and motivates the development of new frameworks for maintaining correctness and security for computations in the many-party setting.

Networked or decentralized computing involves a reliance on a greater number of components, requiring system designers to take into account the potential failures of some of these components. Systems deployed at scale should be resilient to failures of hardware and software components, as well as adversarial unexpected behavior by some system components caused either inadvertently, by deliberate attempts of participating parties to circumvent the protocol, or as a result of infection by malware. Indeed, the recent adoption of blockchain computing establishes a massively distributed computing environment in which anonymous participants have no rationale to trust one another, and often have direct financial incentives to circumvent the system.

The greater availability of large datasets—including those containing both personal information and information about relationships between individuals, as in a social network—

raises major challenges for protecting individual privacy. The existence of such datasets allows sophisticated analyses to be performed, but privacy concerns arise from the publication of analyses based on personal information. However, large datasets also offer the potential to release accurate population statistics while concealing the data belonging to each individual.

This thesis describes a series of problems that focus on issues of security, robustness, and privacy for computations involving many participants. The problems studied span the fields of cryptography, differential privacy, and distributed algorithms. Broadly speaking, we will be concerned with the following goals.

Selective release of information. How can we release some information while provably preserving the secrecy of other information? What aggregate information is safe to reveal without compromising individual privacy, and how accurately can we do so? These questions underly key objectives of differential privacy and cryptography, which are concerned with the disparate but conjoined goals of understanding what information is desirable to release, and how it is possible to reveal only the evaluation of a particular function and no additional information.

Robust computation in network settings. How can we design protocols that are resilient to the failure or malicious behavior of components or parties? How can systems of distributed agents collaboratively perform tasks with only limited and possibly unpredictable communication capabilities? How can agents with only local information about the state of a system coordinate to respond to adversarial events? Understanding the capabilities and limitations of collections of agents in the presence of failures or corruptions is integral to designing resilient protocols and systems in both cryptography and distributed computing.

Overview of this thesis. This thesis is based on a collection of some of the works [Sea16, KRS16, GOSS18, PS19, SU19] I authored or coauthored during my PhD.

Part I introduces a problem of stability in the context of distributed computing. In this setting, a collection of agents with restricted memory and limited ability to communicate are tasked with maintaining a consistent population size in the presence of an adversary that may continually add or remove agents at some bounded rate. Chapter two describes this problem in detail, and presents a protocol that maintains a population close to the target value N, and is robust to nearly $N^{1/4}$ adversarial insertions or deletions in each round of communication.

Part II describes two problems in cryptography that deal with settings of many users. The first, in chapter three, explores the problem of secure multiparty computation in a many-party setting in which infrastructure facilitates rapid secure communication between some pairs of parties, a scenario that could be realized either with precomputation of correlated randomness or through use of special-purpose hardware. Given a network specifying pairs of parties that have efficient pairwise secure computation capabilities, we study when it is possible securely to compute general multiparty computations among all parties in the network.

The second, in chapter four, presents new definitions and constructions for ring signatures that provide more fine-grained control over the capabilities and anonymity guaranteed by the signature scheme. Ring signatures are an anonymity-providing variant of digital signatures, motivated in part by whistleblowing scenarios, in which a signature is produced with respect to a "ring" or set of participants. Informally, they guarantee that the signature was produced by some member of the ring, while also ensuring that parties outside the ring cannot determine which member of the ring was responsible for producing the signature. Chapter four presents new definitions and constructions of ring signature schemes that provide additional guarantees for whether a member of the ring is able to demonstrate authorship or non-authorship of a signature.

Part III explores differentially private release of information in network settings. An algorithm is differentially private if its output distribution is close in a particular sense on any pair of neighboring databases, or databases that are identical except for the data associated with a single individual in the dataset. Chapter five explores the question of privately releasing shortest path and distance information while preserving the privacy of edge weights, for a weighted graph whose topology is publicly known. The chapter introduces the private edge weight model, provides algorithms and lower bounds for privately releasing approximate shortest paths, and provides algorithms for private all-pairs distances in trees, graphs of bounded weight, and general graphs.

Chapter six considers the established but challenging setting of node differential privacy, in which neighboring databases may differ on the set of edges incident to any single vertex, and explores the problem of privately and efficiently releasing the approximate edge density. The chapter presents efficient optimal and near-optimal algorithms for private edge density estimation for two special classes of graphs, graphs whose vertex degrees are contained in an interval of bounded size and Erdős-Rényi graphs.

The remainder of the introduction provides an overview of the results described in each of these sections.

1.1 Distributed protocols robust to population dynamics

The first part of this thesis introduces a coordination problem called the *population stability problem.* The problem is inspired in part by the ability of complex biological systems composed of a multitude of memory-limited individual cells to maintain a stable population size in an adverse environment. Such biological mechanisms allow organisms to heal after trauma or to recover from excessive cell proliferation caused by inflammation, disease, or normal development.

In this problem, a system of agents each with limited memory and communication, as well as the ability to replicate and self-destruct, is subjected to attacks by a worst-case adversary that can at a bounded rate (1) delete agents chosen arbitrarily and (2) insert additional agents with arbitrary initial state into the system. The goal is perpetually to maintain a population whose size is within a constant factor of the target size N.

Chapter 2 describes a population stability protocol in a communication model that is a synchronous variant of the population model of Angluin *et al.* [AAD$^+$04]. In each round, pairs of agents selected at random meet and exchange messages, where at least a constant fraction of agents is matched in each round. This protocol uses three-bit messages and slightly more than $\log^2 N$ states per agent and is robust against an adversary that can both insert and delete agents.

Theorem 1.1.1 (Informal; see Theorem 2.1.1). *For any $\alpha > 0$ and $\Sigma(N) \in \omega(\log^2 N)$, there exists a population stability protocol with three-bit messages and $\Sigma(N)$ states per agent that is robust to $K = O(N^{1/4-\alpha})$ adversarial insertions or deletions per round and maintains the population size between $(1-\alpha)N$ and $(1+\alpha)N$ with overwhelming probability.*

The protocol relies on a novel coloring strategy in which the population size is encoded in the *variance* of the distribution of colors. Individual agents can locally obtain a weak estimate of the population size by sampling from the distribution, and make individual decisions that robustly maintain a stable global population size.

1.2 Infrastructure for secure multi-party computation

The second part of the thesis is concerned with cryptographic protocols for many-party settings. We first consider the setting of secure multiparty computation (MPC), in which a set of N parties wish to compute a joint function of their inputs $x_1, \ldots, x_N$, revealing only the final answer and no other information about the individual inputs. Protocols for secure MPC for general functions are well established [Yao86, GMW87, BGW88, CCD88], but in the dishonest-majority setting, known protocols require relatively expensive public-key operations.

In Chapter 3, we study the possibility of using an infrastructure to improve the efficiency of MPC by avoiding the use of public-key cryptographic computations in the online phase. We propose an infrastructure based on oblivious transfer (OT), which would consist of OT channels between some pairs of parties in the network that could be established through precomputation of correlated randomness shared between pairs of parties, or using special-purpose hardware or other means. We devise information-theoretically secure protocols that allow additional pairs of parties to establish secure OT correlations in the presence of a dishonest majority. The main technical contribution is an upper bound that matches a lower bound of Harnik, Ishai, and Kushilevitz [HIK07], who studied the number of OT channels necessary and sufficient for MPC. In particular, we characterize which n-party OT graphs G allow t-secure computation of OT correlations between all pairs of parties, showing that this is possible if and only if the complement of G does not contain the complete bipartite graph $K_{n-t,n-t}$ as a subgraph.

1.3 Fine-grained control for ring signatures

Ring signatures, introduced by Rivest, Shamir and Tauman [RST01], are a variant of digital signatures which certify that *one among a particular set* of parties has endorsed a message while hiding *which* party in the set was the signer. Ring signatures are designed to allow *anyone* to attach anyone else's name to a signature, as long as the signer's own name is also attached.

But what guarantee do ring signatures 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? Prior security definitions for ring signatures do not give a conclusive answer to this question: under most existing definitions, the guarantees could go either way. That is, it is consistent with some standard definitions that a non-signer might be able to *repudiate* a signature that he did not produce, or that 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. Any of these guarantees might be desirable. For instance, a whistle-blower might have reason to want to later claim an anonymously released signature, or a person falsely implicated in a crime associated with a ring signature might wish to denounce the signature that is framing them and damaging their reputation. In other circumstances, 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. In any case, a *guarantee* one way or the other seems highly desirable.

In Chapter 4, we formalize definitions and give constructions of the new notions of *repudiable*, *unrepudiable*, *claimable*, and *unclaimable* ring signatures. Our repudiable construction is based on VRFs, which are implied by several number-theoretic assumptions (including strong RSA or bilinear maps); our claimable construction is a black-box transformation from any standard ring signature scheme to a claimable one; and our unclaimable construction is derived from the lattice-based ring signatures of Brakerski and Kalai [BK10], which rely on the hardness of the short integer solutions problem (SIS). Our repudiable construction also provides a new construction of standard ring signatures.

1.4 Differential privacy for network data

The final part of the thesis is concerned with privacy-preserving data analysis for network-structured data. Differential privacy, defined by Dwork, McSherry, Nissim and Smith [DMNS06], requires that the output of an algorithm provides little advantage, measured by privacy parameters ε and δ, for distinguishing between *neighboring databases*, which are thought of as databases that differ on the contribution of one individual. In the most typical setting, each row of the database consists of the data of a single individual, so a pair of databases are neighbors if they are identical except for a single row. Complications arise in differential privacy for databases corresponding to graphs, since the graph structure generally does not decompose naturally into the data contributions of each individual, and changing which edges are incident to a single vertex may change the local topology of every other vertex as well.

Under *node differential privacy*, individuals correspond to vertices, and neighboring databases differ only in the set of edges incident to a single vertex. This model is natural for modeling social networks and many other settings in which vertices correspond to individuals and it is desirable to hide the characteristics of any single individual. However, many graph properties are highly sensitive to changes in the neighborhood of a single vertex, and node-differentially private algorithms are known only for a relatively small set of graph properties. A weaker but still interesting guarantee is provided by *edge differential privacy*, in which neighboring databases may differ only the presence or absence of a single edge. Chapters 5 and 6 are concerned with two problems concerning privacy-preserving analysis of graphs, the latter in the setting of node differential privacy and the former in a setting in which neither node-level privacy nor edge-level privacy is suitable.

1.4.1 Shortest paths and distances under differential privacy

Shortest paths and distances between the vertices of a graph are are among the most fundamental problems in graph algorithms. However, neither problem seems amenable to privacy-preserving solutions under node or edge differential privacy. This is because the addition or removal of even a single edge may drastically change connectivity and distances, and may even disconnect the graph. For shortest paths the situation is even worse, since releasing any valid path, however short, reveals a set of edges contained in the graph, violating both privacy definitions.

Chapter 5 introduces a different model for differentially private analysis of weighted graphs that captures desirable privacy constraints for many applications and in which shortest path and distance queries can be performed privately. In this model, the graph topology $(\mathcal{V}, \mathcal{E})$ is assumed to be public, and the private information consists only of the edge weights $w : \mathcal{E} \to \mathbb{R}^+$. Two weight functions w, w' are considered to be neighboring if they have ℓ_1 distance at most one. This can model, for instance, congestion patterns on a known system of roads, where the presence or absence of a single car can have a bounded influence on the traffic along a road or set of roads.

In this setting we can study the problems of privately releasing a short path between a pair of vertices and of privately releasing approximate distances between all pairs of vertices. For the problem of privately releasing a short path between a pair of vertices, we prove a lower bound of $\Omega(n)$ on the additive approximation error for an n-vertex graph, for fixed privacy parameters ε, δ. We provide a differentially private algorithm that matches this error bound up to a logarithmic factor and releases paths between all pairs of vertices, not just a single pair. The approximation error achieved by our algorithm can be bounded by the number of edges on the shortest path, so we achieve better accuracy than the worst-case bound for pairs of vertices that are connected by a low-weight path consisting of $o(n)$ vertices.

For the problem of privately releasing all-pairs distances, we show that for trees we can release all-pairs distances with approximation error $O(\log^{2.5} n)$ for fixed privacy parameters ε, δ. For arbitrary bounded-weight graphs with edge weights in $[0, M]$ we can release all distances with approximation error $\tilde{O}(\sqrt{nM})$.

1.4.2 Private estimation of edge density

In Chapter 6 we consider one of the most basic queries about a graph, namely the edge density $|E|/\binom{n}{2}$. It is straightforward to release the edge density of an arbitrary graph under node differential privacy with error roughly $O(1/(\varepsilon n))$, which is optimal for worst-case graphs. However, it is still possible to achieve better accuracy over a restricted class of graphs while still guaranteeing privacy in the worst case. We study two particular families of graphs, Erdős-Rényi random graphs and arbitrary graphs satisfying the property that all vertices have close to the same degree.

A recent result of Borgs, Chayes, Smith and Zadik [BCSZ18] shows that the parameter p of an Erdős-Rényi graph $G(n, p)$ can be estimated privately with error $\tilde{O}(1/n + 1/(\varepsilon n^{1.5}))$, which is believed to be optimal. However, the result is based on general Lipschitz extensions and consequently yields an exponential-time algorithm.

We give a simple, computationally efficient algorithm for privately estimating the parameter of an Erdős-Rényi graph (and also the edge density) with close to optimal accuracy. Our algorithm has error $\tilde{O}(1/n + 1/(\varepsilon^2 n^2))$. In particular, for $\varepsilon \geq 1/\sqrt{n}$, we achieve error $\tilde{O}(1/n)$, which (up to logarithmic factors) is optimal and matches both the nonprivate accuracy for the problem and the accuracy of the exponential-time algorithm of Borgs *et al.* More generally, we give an optimal, computationally efficient, private algorithm for estimating the edge-density of any graph whose degree distribution is concentrated in a small interval.

Part I

Distributed algorithms

Chapter 2

Population stability: regulating size in the presence of an adversary

This chapter is based on joint work with Shafi Goldwasser, Rafail Ostrovsky, and Alessandra Scafuro [GOSS18].

2.1 Introduction

A single fertilized mouse egg and human egg develop into organisms with vastly different numbers of cells. How do cellular mechanisms regulate the number of cells in complex biological systems? In order to function in adverse environments, organisms must maintain stability and be able to recover from unplanned circumstances. For example, a lizard that loses its tail can grow a new one, and internal organs require mechanisms to recover from cell loss caused by injury or from execessive cell proliferation due to development or disease. But how do individual cells know how to respond in order to reestablish the desired population size?

Regulation of population size may be achieved through a combination of internal programs running within each cell and intercellular communication. One approach could be for individual cells to count the population using a distributed protocol. But an interesting question is how to control the population size if each cell lacks sufficient memory to count. Understanding the mechanisms for regulating population size in an adversarial environment, in light of memory constraints of individual agents, is a natural computational question.

In this work, we study the problem of robustly maintaining a stable population size from the perspective of distributed computing. We introduce a new coordination question that we call the **population stability problem**. Consider a population of agents with the ability to replicate and self-destruct. How can such distributed systems detect and recover from adversarial deletions and insertions of agents so as to maintain the desired population size?

Our focus is on systems that consist of huge numbers of agents, where each agent individually has very limited memory and connective capability and can directly communicate with only a few other agents in the system. We model communication using a synchronous variant of the population model of Angluin et al. [AAD+06, AAE07]. In each round, a constant fraction of agents is matched at random and can exchange messages, where the matched agents are chosen independently in each round. Population size must be maintained in this setting in the presence of an adversary that observes the entire state of the system and can continually delete or insert agents. We describe the model in more detail below.

The population stability problem augments a growing body of work that uses the language and ideas of distributed computing to model biological systems consisting of a collection of resource-constrained components that collectively accomplish complex tasks. Naturally, we do not claim direct relevance of our results to biological systems due to potential modeling differences. Regardless, the population stability problem makes sense in any system consisting of individual components with the ability to reproduce.

2.1.1 Contributions

Our main contributions are as follows.

A New Problem in Distributed Computing: The Population Stability Problem. We introduce a new problem in distributed computing. A population of N memory-constrained agents (i.e. processors with the ability to reproduce and self-destruct) is subjected to adversarial attacks. Whereas many attacks can be envisioned, we consider a worst-case adversary that can delete or insert agents at a bounded rate. The goal is to maintain a stable population size within a small multiplicative factor of the original size N. This problem appears fundamentally different from the classical problems of distributed computing, such as consensus, leader election, majority, common coin flipping, or computing general functions of the joint state of the parties.

Models for Communication and the Adversary. The communication model we consider is a synchronous variant of the population model of [AAD+06, AAE07]. That model was designed to represent sensor networks consisting of very limited mobile agents with no control over their own movement and whose goal is to compute some function of their inputs or evaluate a property of the system. Whereas [AAD+06] assumed that pairs of agents can communicate via pairwise interactions as scheduled by a uniformly random matching process, we assume in addition that agents are synchronized and interact with one another in rounds. Within each round, at least a γ fraction of agents participates in pairwise interactions, again scheduled uniformly at random. The agents additionally have the ability to self-destruct and to reproduce by producing a second identical copy of themselves.

It is clear that we cannot allow the adversary to delete most or all of the agents in a single round, since maintaining a stable population size in the presence of such an adversary is impossible. Consequently, we give the adversary a budget of K alterations to perform in each round, where an alteration consists of removing, inserting or modifying the memory of a single agent. We allow the adversary to observe the memory contents of *every* agent before determining its alterations for a round. Both γ and K are parameters of the model. The model is described in detail in Section 2.2.

Protocol for Population Stability. We present a protocol with three-bit messages requiring polylog(N) states (i.e. $\Theta(\log\log N)$ bits of memory) per agent that tolerates $K = N^{1/4-\varepsilon}$ worst-case insertions or deletions in each round, for any constant $\varepsilon > 0$. Formally, our main theorem is the following.

Theorem 2.1.1. *Let $\alpha, \gamma, \varepsilon$ be positive constants, where γ is a lower bound on the fraction of agents that is matched in each round. Then there exists a population stability protocol with three-bit messages and* $\mathrm{polylog}(N)$ *states per agent and guaranteeing that if the adversary inserts or deletes at most $K = O(N^{1/4-\varepsilon})$ agents in each round, then with all but negligible probability the population will remain between $(1-\alpha)N$ and $(1+\alpha)N$ for any polynomial number of rounds.*

New Techniques. The main idea employed in our construction is to *color* the agents with the values 0 and 1 in such a way that information about the population size is encoded in the distribution of colors. That is, given a set of agents with assigned colors in $\{0, 1\}$, consider the distribution specified by choosing an agent at random and observing its color. We are able to assign colors to agents in such a way that approximate population size is encoded in the *variance* of this distribution. Subsequently, each individual agent locally computes a very weak estimator of whether the variance is too large or too small, and makes a local decision of whether to reproduce or self-destruct. Although each individual agent's estimate is noisy, we show that in the aggregate, the local decisions are globally able to maintain a stable population size even in the presence of a powerful adversary.

We discuss the model and results further in Section 2.1.2. We provide a more in-depth overview of our techniques in Section ?? below. The model is described in further detail in Section 2.2. In Section 2.3 we provide a full description of the protocol, and in Section 2.4 we present the analysis.

2.1.2 Discussion and extensions

Challenges. The first obstacle we must confront in designing a protocol for population stability is the memory constraint. Each agent does not have sufficient memory to store a unique identifier or to count to the population target N. Yet collectively, they must have a good approximation of N and must individually decide whether to replicate if the population is too low or to self-destruct if the population is too high. Making their task even more challenging, these memory-constrained agents must correct deviations in the size of the population and make their decisions *while* the adversary is acting.

A second major challenge is that the adversary is very powerful. Although the number of agents inserted or deleted in each round is bounded, the adversary can observe the complete state of the system[1] before deciding on its actions, and may insert agents of arbitrary initial state. These capabilities present difficulties for many standard techniques and abstractions used in distributed computing. For instance, many protocols are based on leader election, where a single leader processor is chosen to direct or facilitate the task at hand. However, since our adversary is able to observe the state of every agent, the adversary can simply wait for a leader to be chosen and then delete it. Furthermore, even without observing the internal memory of agents, the adversary could insert many additional agents that are all identical in state to the leader. Indeed, since agents can be in one of only polylog(N) distinct states and the adversary is able to insert $K = N^{1/4-\varepsilon} \in \mathsf{poly}(N)$ agents per round, the adversary can even insert many copies of *every possible* agent type in each round. Consequently any approach that relies on the existence of agents of unique or extremely special state, such as leader election, seems doomed to failure. This appears to render ineffectual a large part of the distributed computing toolbox.

Adversarial insertions. Recall that we allow the adversary to insert new agents with arbitrary initial state. Starting from that internal state, we assume that the inserted agents execute the same protocol as honest agents. We could instead consider an even stronger adversary that inserts agents running arbitrary malicious protocols specifying their subsequent behavior. However, the population stability problem as described above is clearly impossible in the face of this stronger adversary, since our model does not include the ability to destroy agents that do not cooperate. A malicious agent can simply ignore all interactions with other agents and replicate itself at every opportunity. Such malicious agents would quickly replicate themseves out of control, rapidly exceeding the population target.

[1]That is, the adversary has the ability to read the memory contents of every agent.

One may consider a different model that allows agents not only to self-destruct but also to remove other agents it encounters. In such a setting, our protocol can be extended to achieve population stability even if the adversary is allowed to insert agents that execute arbitrary malicious programs, as long as there is a bound on how frequently malicious agents can replicate and an agent is able to detect when it encounters an agent whose program is different from its own. However, that setting is not the focus of this work.

Correction and detection. The objective in the population stability problem is to maintain a population size that is close to a target value N, and to correct the population if it deviates too far from this target. A closely related problem is that of simply *detecting* whether the population has deviated too far from the target or whether it has exceeded some threshold, objectives that seem very similar to the problem of approximate counting [Mor78]. It would be natural to try to solve our problem by first detecting whether the population is too large or too small and then correcting appropriately. With a weaker adversary that can only delete agents but not insert additional ones (and additionally is oblivious to the internal states and coin flips of agents), this approach can be made to work using approximate counting techniques. In the adversarial model considered here, constructing approximate counters and detecting changes in population size are interesting open questions.

Synchrony. In this work we study a synchronous model, where all agents communicate and perform updates in rounds. As has commonly been the case across distributed computing, it is natural to study a new problem first in a synchronous setting to distill key ideas and techniques before adding additional complications in an asynchronous setting.

We note, however, that synchronization is far from an unreasonable assumption in biological systems. Indeed, many multicellular systems do achieve synchrony either through regular external stimuli such as sunlight or through chemical control mechanisms. For instance, heart cells maintained in culture were able to achieve a high degree of synchronization of their rhythmic contractions [JMPT87]. Neuronal cells too exhibit highly synchronized behavior. Even when grown in culture, specialized neuronal cells show the capacity to synchronize the release of particular hormones at regular time intervals [DLECW92]. Bacterial populations have also been shown to have the capability of producing coordinated oscillations [MO09, DMPTH10].

Nonetheless, an extremely natural and interesting question is how to solve the population stability problem in a setting without synchrony or with only partial synchrony. For instance, one could consider a setting where agents have clocks that have bounded drift relative to one another. Related to this is the typical random scheduler setting of population protocols [AAE07], in which a single pair of agents at a time is chosen to interact and update state. By a concentration argument, this process allows agents to maintain clocks that do not drift too quickly relative to one another.

While the construction in this paper requires synchrony, there are some known techniques in the population protocol literature for maintaining approximate synchronization in a non-synchronous setting; see, for instance, the recent work of [AAG18]. A natural extension is to show whether our techniques can be combined with synchronizers to achieve population stability in such settings.

Alternate communication models. Another very interesting question is to explore the population stability problem under a different communication model. In this paper, pairs of agents that communicate are chosen independently at random in each round. Alternatively, one could consider settings in which the neighbors of an agent are consistent over time, perhaps reflecting underlying geometric constraints.

One natural approach is to use a fixed sparse communication graph, for instance an expander. However, modeling problems arise in determining how connectivity changes upon agent replication, insertion, or deletion. In various settings along these lines, it is straightforward for the adversary to disconnect the communication graph and consequently to violate population stability. An alternate approach could be to associate agents with points in $\mathbb{R}^d$, and to allow each agent to communicate with a small number of the nearest other agents.

Population stability in the high-memory setting. We note that in the absence of memory constraints, there is a trivial protocol both for approximate counting and for the population stability problem if the adversary can only delete and not insert new agents. Each agent simply flips N coins to generate a unique identifier $\mathsf{id} \in \{0,1\}^N$. For an interval of $O(\mathsf{poly}\log(N))$ rounds each agent broadcasts the set of identifiers it has received so far. With high probability, all identifiers are unique and agents receive the identifiers of every agent that was alive throughout the interval, so each agent learns a close approximation of the population size and can make a decision of whether to self-destruct, replicate, or neither. However, this protocol relies heavily on agents having very large memory, and does not yield an approach to solve the problem in the low-memory setting.

A note about success probability. Theorem 2.1.1 states that our protocol maintains a stable population for any polynomial rounds with all but negligible probability. Recall that a function is *negligible* if it tends to zero faster than any polynomial. That is, a function $\mathsf{negl}(x)$ is negligible if for any $c \in \mathbb{N}$ there exists an x_0 such that $|\mathsf{negl}(x)| < 1/x^c$ for all sufficiently large $x \geq x_0$.

Throughout this paper, we use the phrases "with high probability" and "with overwhelming probability" to mean with probability $1 - \mathsf{negl}(N)$ for some negligible function negl.

2.1.3 Technical overview

Preliminary attempts We first describe two preliminary attempts at protocols for the population stability problem which, while unsound, will provide useful intuition toward the design of our actual protocol.

Attempt 1: non-interactive leader election As a first attempt in the low-memory setting, consider the following approach, which is based on an idea from [AAE+17]. Each agent flips a biased coin where $\Pr[c = 1] = \frac{1}{N}$, where outcome 1 means that the agent is a leader. For $O(\text{polylog}(N))$ rounds, each agent sends any agent it encounters the bit 0 if its coin was zero and it has not received the message 1, and the bit 1 if its coin was 1 or it has received a 1 from another agent. In the absence of an adversary, this allows every agent to learn whether a 1 was obtained in any of the initial coin flips. The probability of this event differs noticeably depending on whether the population is too small or too large. After repeating to amplify the signal, with high probability the agents can detect if the population is too small or too large and can replicate or self-destruct accordingly. A protocol of this form can be shown to work in the presence of an adversary that can only delete and not insert agents, and is additionally oblivious to the coin flips made by the agents. However, in the adversary model considered here with insertion as well as deletion and full knowledge of the states of agents, the protocol will fail. The adversary can either insert an agent with coin value $c = 1$ in each phase, or else identify the agent or agents with coin value 1 and selectively remove these agents. Consequently the adversary can cause the population to grow or shrink arbitrarily.

This attempt highlights a fundamental difficulty in designing protocols in our adversarial model. The protocol relied on a non-interactive strategy related to leader election, where the presence or absence of a leader could be used to infer the approximate size of the population. However, as we have discussed above, the use of a special state with one or only a few agents of that state (in this case agents with coin value $c = 1$) provides the adversary with an easy avenue of attack, namely the deletion of agents of that state or the insertion of many additional agents of that state. Consequently, constructions of this flavor seem to have little promise in this adversarial setting.

Attempt 2: independent coloring As a next attempt, consider the simple protocol in which each agent flips a fair coin, receiving at random a color $c \in \{0, 1\}$. For each agent, compare the colors of the next two agents encountered. If the colors are equal, then split, and otherwise self-destruct. Observe that if an agent encounters the same agent twice, the colors must be the same, while if an agent encounters two different agents, the colors are independently random. Consequently if the population currently has size m, then the probability of splitting is $\frac{1}{2} + \Theta(\frac{1}{m})$, which is slightly larger than the probability $\frac{1}{2} - \Theta(\frac{1}{m})$ of self-destructing, and so this protocol would cause the population to increase slightly over time. To compensate this, modify the protocol to split only with probability $1 - \Theta(\frac{1}{N})$ if the colors are equal while still self-destructing with probability 1 if the colors are unequal.[2]

Now, the population will stay the same size in expectation if its current size m is equal to the target N, will decrease in expectation if $m > N$, and will increase in expectation if $m < N$. Qualitatively, this is exactly the behavior that we want. However, tending in expectation to correct itself is insufficient for maintaining a stable population. In fact, despite a very weak bias to correct drifts in the population, the signal is overwhelmed by the noise, and the size of the population under this protocol will behave very much like a random walk. Even in the absence of an adversary, this protocol will cause the population to drift extremely far from its initial size.

In some sense the protocol we have just outlined behaves even worse than the empty protocol, in that it fails to maintain a stable population when there is no adversary at all. However, the protocol does have one intriguing feature. It entirely lacks any "special" agent types for the adversary to exploit. Consequently, if we could design a more sophisticated protocol along these lines that could maintain a stable population in the absence of an adversary, we might hope that it could do the same even in the presence of an adversary.

Overview of our protocol

We now describe our actual protocol. At a very high level, the idea behind our protocol is as follows. Through some *coloring process* which we will discuss below, agents are colored with the colors $\{0, 1\}$. After the agents are colored, agents run a step called the *evaluation phase* in which agents make the decision of whether to reproduce or self-destruct. The coloring process and evaluation phase are then repeated indefinitely. We will refer to each iteration of the coloring process followed by the evaluation phase as an *epoch.*

During the evaluation phase, each agent that is matched with another agent in this round compares its own color with the color of its *neighbor* (i.e. the agent to which it is matched). If

[2]Another perspective on why the protocol without this step cannot maintain a stable population is that the protocol run by each agent has no dependence on the population target N. Consequently, if agents are added or removed (either by the adversary, or even as a result of random drift) then the protocol should behave as if those agents were there to begin with and will not correct this deviation in the population.

the two agents have the same color, then the agent will replicate itself with some probability p_{split}. If the two agents have different colors, then the agent will self-destruct. Note that if the coloring process consisted of every agent tossing its own coin, then this would be essentially the same as Attempt 2 above. We will instead employ a more structured coloring process that results in agent colors that are not generated independently at random.

The coloring process will consist of two phases, a (noninteractive) *leader selection phase* and a *recruitment phase.* In the leader selection phase, $\Theta(1/\sqrt{N})$ of the agents will become "leaders"[3] and each leader will choose a random color in $\{0, 1\}$. In the recruitment phase, each leader will identify $\sqrt{N}$ uncolored agents and color each of them with its own color. We note that the leader will not directly encounter each of these $\sqrt{N}$ agents, but will directly color some agents which in turn will color other agents. To begin with, each leader activates the first inactive agent that it encounters, sharing its color with the new agent. Each agent is subsequently responsible for recruiting $\sqrt{N}/2$ inactive agents in the same manner, forming a recruitment tree of depth $\frac{1}{2}\log N$. By delegating the coloring in this manner, the recruitment process can be performed in only $O(\log^2 N)$ rounds.[4]

For each leader, at the end of the recruitment phase, $\sqrt{N}$ agents will have obtained a color based on the original coin toss of that leader. We will refer to these $\sqrt{N}$ agents as the *cluster* associated with the leader, and we will sometimes describe agents as belonging to the same cluster or different clusters. At the end of the recruitment phase and the beginning of the evaluation phase, a constant fraction of the population will have been colored having been recruited into the clusters of the various leaders, roughly half with color 0 and the other half with color 1.

Consider a particular agent in the evaluation phase. If the agent meets another agent from the same cluster, then they necessarily have the same color. If the agent meets an agent from a different cluster, then their colors are independently random. Consequently, if the current population size is m, then with probability $\frac{1}{2} + \Theta(\frac{\sqrt{N}}{m})$ the two agents will have

[3]That is, each agent will become a leader with probability $\Theta(1/\sqrt{N})$.

[4]In order to achieve constant message size, our full protocol will be slightly different and will use additional rounds for the recruitment process.

the same color, and with probability $\frac{1}{2} - \Theta(\frac{\sqrt{N}}{m})$ the two agents will have different colors. We can choose the splitting probability $p_{\mathsf{split}} = 1 - \Theta(1/\sqrt{N})$ so that the expected change in population is zero for $m = N$. For $m \gg N$ the expected change in population will be negative, and for $m \ll N$ the expected change in population will be positive. Moreover, unlike in Attempt 2 above, which behaved similarly to a random walk, here the effect is strong enough to maintain the population in a small interval around the target value N, with all but negligible probability.

Moreover, the adversary can do little to influence the result of the protocol. For a population size $m = \Theta(N)$, the number of leaders selected is $\sqrt{N}$, and so the standard deviation of the number of leaders with each color is roughly $N^{1/4}$. Consequently, an adversary that can insert or delete $o(N^{1/4})$ agents can do little to influence the distribution of colors. Even if the adversary selectively inserts or deletes agents that are leaders and have a particular color, the effect of the adversary on the distribution of colors is dominated by the random deviation of the sampling process. Unlike in Attempt 1 above, there are many leaders, and so the adversary is unable to delete enough of them or to insert enough additional leaders to overwhelm the protocol. As we will show, the adversary cannot cause substantial deviations in the size of the population.

Note that one strategy the adversary may attempt is inserting agents that do not know the correct round number within the epoch. In the protocol described so far, there is no mechanism for detecting and correcting this, and so over many rounds, adversarial insertions may lead to a population of agents attempting to execute different portions of the protocol. In order to address this, agents can exchange which round they are in, and self-destruct upon encountering an agent that is in a different round of the epoch. This results in the self-destruction of any agent with the wrong round number as soon as it encounters an agent with the correct round number. A corresponding number of correct agents are also destroyed, but we will show that the number of correct agents removed in this manner is sufficiently small.

As discussed briefly in Section 2.1.1, we can think of this protocol as encoding the population size in the variance of a distribution and then sampling from this distribution to obtain a weak estimate of the variance. Since the variance of the fraction of successes in many independent Bernoulli trials decreases as the number of trials increases, if the number of leaders is larger, then the fraction of colored agents with color 0 will be more closely concentrated around 1/2. On the other hand, if the number of leaders is smaller, then we expect the fraction of colored agents with color 0 to be farther from 1/2. Since the expected number of leaders is proportional to the current size of the population, an approximation to the population size is encoded in the fraction of agents of each color. Consider the distribution obtained by selecting an agent at random and reading its color. Comparing the colors of two agents serves as a very weak estimate of the variance of this distribution, while aggregating the results of many agents' individual choices of whether to replicate or self-destruct serves to amplify the accuracy of this estimate.

Achieving constant-size messages. The protocol described above involves messages of size $\Theta(\log \log N)$ bits, essentially as large as the entire memory of an agent. We now outline how to modify the protocol to use constant-size messages. The only large portions of the messages described so far consist of the current round in the epoch and the depth in the recruitment tree, each of which can be encoded in $\Theta(\log \log N)$ bits.

The current round in the epoch is sent to prevent the adversary from confusing the protocol by inserting agents with the wrong round number. However, rather than sending the exact round, we will instead send the single bit specifying whether or not the agent is currently in the evaluation round. If an agent is entering the evaluation round and its neighbor is not, then both agents will self-destruct. We can show that this suffices to maintain the invariant that a large majority of the agents are in the same round of the epoch.

The depth of the recruitment tree is sent to allow each leader to induce the recruitment of the correct number of agents. Note that we cannot simply recruit for $\frac{1}{2} \log N$ rounds, since recruiting agents may encounter other agents that are already colored and cannot recruit them. However, we can slow down the recruitment process to allow the depth in the recruitment tree to be determined as a function of the round number. To do this, we recruit for $\Theta(\log^3 N)$ rounds, divided into $\frac{1}{2}\log N$ subphases of $\log^2 N$ rounds each.[5] In a single subphase, a recruiting agent will recruit only the *first* inactive agent it sees even if it encounters many inactive agents in the subphase. This allows agents to determine their depth in the recruitment tree based on the round in which they were recruited. Since a subphase consists of $\omega(\log N)$ rounds, we will show that an inactive agent will be encountered with high probability.

This yields a population control protocol with constant-size messages, since messages consist of four binary values, namely an agent's color, whether or not it is active, whether or not it is recruiting, and whether or not it is currently in the evaluation round. In the analysis we will see how to achieve the same result with only three-bit messages.

2.1.4 Related work

Population Protocols. The population protocol model was introduced by Angluin et al. [AAD+04, AAD+06]. In this model a collection of agents, which are modeled by finite state machines, move around unpredictably and have pairwise interactions. The original definition considers a worst-case environment/scheduler, while later formulations [AAE07] consider the case where each interaction occurs between a pair of agents chosen uniformly at random. In a population protocol, agents start with an initial configuration, and the goal is to jointly compute a function of this input. Previous works have tried to identity the class of functions that can be computed in such a model [AAER07], and the tradeoffs between the resources need to do so (e.g. [AAE+17]). In these works, the agents are always active throughout the execution of the protocol.

Another line of work expands the population model to the case in which agents can crash or undergo transient failures that corrupt their states. Delporte-Gallet et al. [DFGR06] consider a setting in which agents must compute a function of their inputs in presence of such failures. They construct a compiler that takes as input a protocol that works in the

[5]We actually only require that subphases are $\omega(\log N)$ rounds, so it is sufficient to recruit for $\omega(\log^2 N)$ rounds.

failure-free model, and outputs a protocol that works in the presense of failures as long as modifying a small number of inputs does not change the function output. Angluin et al. [AAFJ08] incorporated the notion of self-stabilization into the population protocol model, giving self-stabilizing protocols for some classical problems such as leader election and token passing. They focus on the goal of stably maintaining some property such as having a unique leader or a legal coloring of the communication graph.

Unlike these works, in our work agents have the ability to reproduce and self-destruct, and the goal of maintaining a consistent population size must be carried out in the presence of an adversary with the corresponding capability to insert and delete agents.

Approximate Counting. The problem of maintaining the population size of a collection of memory-constrained agents is related to the problem of counting N items when the available memory of the agents is less than $\log N$ bits. Approximate counters were introduced by Morris [Mor78] as technique to accurately approximate a value N using only $\Theta(\log \log N)$ bits of memory. Techniques for approximate counting in the population model were developed in [ABBS16, AAE+17].

A sequence of works by Di Luna et al. [LBBC14b, LBBC14a] consider the problem of estimating the size of a network of agents that communicate according to a dynamic connection graph, in presence of an adversary that can add and remove *edges* in the graph.

Cellular Automata. Cellular automata were proposed by von Neumann [vN51] as a model to reason about artificial self-reproduction. A cellular automaton consists of a regular grid of cells, each assuming one of a finite number of states. Over time, the states of cells change according to some fixed rule (e.g. a mathematical function) that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood. Conway's Game of Life [Gar70] is a cellular automaton that works with a simpler set of rules than von Neumann's rules, and was shown to be Turing-complete by Berlekamp, Conway and Guy [BCG04]. Cook [Coo04] proved that rule 110 (a binary, *one*-dimensional cellular automata) is Turing-complete.

Our setting is crucially different from the cellular automaton setting, since agents in our model do not simply change state but can be deleted from the system during the computation. Another difference between our setting and the cellular automaton setting is that we consider an *adversarial* model whereas in cellular automata cells deterministically change state. In some sense, in Conway's game, death "plays by the rules" while in our game death is sudden and unpredictable.

Dynamic Environments. A recent work of Goldreich and Ron [GR17] considers environments that evolve according to a fixed local rule. They define an *environment* as a collection of small components of a large system which interact in a local level, and change state according to a fixed rule. As an example, they focus on the model of a two-dimensional cellular automata. They ask how many queries a global observer must make about local components in order to test whether the evolution of the environment obeys a fixed known rule or to predict the state of the system at a given time and location. Although their work seems very different than ours, it bears some intellectual similarity in seeking information about a global property of the system from local information. However, whereas in [GR17] a global observer who can query a limited number of individual cells asks "does the global system obey a specific evolution rule," in our case individual agents need to decide locally what they should do to maintain the overall global property of population size.

Self-Stabilization. Also related to our question is the self-stabilization problem introduced by Dijkstra [Dij74]. Given a system that starts in an arbitrary state, the goal of a stabilization algorithm is to eventually converge to the correct state. In this setting, however, deletion of system components is not considered. Super-stabilization [DH97] is the problem of achieving self stabilization in dynamic networks, that is, network where nodes are dynamically added and removed. While this setting is closer to ours, super-stabilization algorithms make additional assumptions about the system, such as that each node in the system is uniquely identified.

Distributed Algorithms Explaining Ant Colony Behaviors. A single ant has very limited communication and processing power, yet collectively a colony of ants can perform complex tasks such as consensus decision-making, leader election, and navigation. In [CDLN14]

Cornejo et al. give a mathematical model for the problem of task allocation in ant colonies, and propose a very efficient protocol for satisfying this task. One of the main goals of their paper is to provide a formal model enabling the comparison of the various task allocation algorithms proposed in the biology literature. Similarly, in [GMRL15] Ghaffari et al. use techniques from distributed computing theory in order to gain insight into the *ant colony house-hunting problem*, where a set of agents need to identify potential nests, evaluate the quality of candidates, and reach consensus in a distributed manner.

2.2 The Population Stability Problem

As discussed above, the population stability problem is concerned with a system of agents with bounded memory and the ability to reproduce and self-destruct. The system is subjected to adversarial attacks that delete or insert processors. The objective is to maintain a stable population size despite these adversarial attacks. In this section we give a formal description of the problem.

Parameters. The population stability problem is parameterized by the initial number of agents N, the number of distinct memory states M each agent can be in, the number of alterations K the adversary is allow to make in each round (where an alteration consists of removing or inserting an processor), a value α specifying how tightly concentrated the population size must remain around the target value N, and a value γ specifying a lower bound on the fraction of processors that are matched with other processors in each round. Below we describe each of these components of the problem.

We note that one could consider separate parameters for the number of adversarial deletions and insertions, allowing the adversary to make a different number of each. In this paper we will consider both to be bounded by a single parameter K.

Agents. We consider agents with bounded memory. Each agent can be in one of M possible states, so we can think of agents as having $\log M$ bits of memory. Agents can communicate by message-passing as specified by the connectivity structure of the system, which will be discussed further below. In our setting we will have $M \ll N$, so each individual agent has insufficient memory to count the total population, to posses a unique ID, or to address messages to a particular recipient. Each agent has the ability to flip unbiased coins, to split into two identical agents, or to self destruct. That is, in any round an agent may choose to split into two daughter agents which both inherit specified state from the parent agent, or it can decide to delete itself from the system.

Connectivity. As discussed above, we consider a *synchronous* version of the population model of Angluin et al. [AAD$^+$04, AAE07], where we assume the existence of a global clock. We assume that the pairs of agents that are able to communicate in each round are selected by choosing a random matching of at least a γ fraction of surviving agents. We think of the parameter γ as a constant (e.g. $\gamma = 1/4$). That is, each agent is matched with at most one other agent (that we call its *neighbor*) in each round, and there is no consistency from round to round, since connectivity in different rounds is determined by sampling independently random matchings. The schedule of these matchings is unknown to the adversary in advance.

Adversary. We consider a worst-case, computationally unbounded adversary that can arbitrarily choose which agents to delete in each round and can insert agents with arbitrary state in each round. The adversary also can observe the entire history of agent interactions, including the memory contents of every agent. While the initial state of inserted agents is determined by the adversary, the newly inserted agents are assumed to follow the protocol (that is, the agents introduced by the adversary do not behave maliciously).

Objective. The goal in the population stability problem is to maintain a number of agents within a small interval $[(1-\alpha)N, (1+\alpha)N]$ around the initial population size N. That is, initially the system consists of N agents. In each round some agents may be removed or inserted by the adversary, and some agents may decide to replicate or to self-destruct. Let N_i denote the number of agents in the system after the ith round. The adversary wins in round i if $N_i \notin [(1-\alpha)N, (1+\alpha)N]$ at the end of the round. We say that protocol Π is a population stability protocol if for any polynomial p and any adversary, the probability that the adversary wins in at most $p(N)$ rounds is negligible in N.

2.3 The protocol

We now provide a formal specification of the main protocol (Algorithm 1). Agents continually run the protocol throughout their lifetime. We will think of time as partitioned into epochs of $T = \frac{1}{2}\log^3 N$ rounds. Each epoch consists of three phases, the leader selection phase, the recruitment phase, and the evaluation phase. The recruitment phase consists of $\frac{1}{2}\log N$ subphases each consisting of roughly $T_{\mathsf{inner}} = \log^2 N$ rounds.[6] We will elaborate on each of these phases below.

Recall that agents have the ability to toss coins, to send and receive a message upon encountering another agent, to reproduce by splitting into two identical copies of itself, and to self-destruct. These capabilities are notated by the following functions. We denote flipping an unbiased coin by $x \overset{\$}{\leftarrow} \{0,1\}$. Agent splitting and death are implemented in commands $\textsc{Split}()$ and $\textsc{Die}()$. Finally, the command $Z := \textsc{Communicate}(X)$ sends message X to the neighboring agent in the present round, if any, simultaneously receiving in response message Z.[7] If the agent is unmatched in this round and has no neighboring agent, then the return value is assumed to be $\perp$. In the protocol below, messages will consist of four boolean values $(\mathsf{inEvalPhase}, \mathsf{active}, \mathsf{color}, \mathsf{recruiting})$. Upon receiving message Z, the value of each of these variables can be accessed by writing $Z.\mathsf{inEvalPhase}$, $Z.\mathsf{active}$, and so forth. If $Z = \perp$ then we will follow the convention that each of these components will also have value $\perp$.

The main variables that describe the state of an agent are $\mathsf{round} \in [0, T-1]$ and four boolean variables $\mathsf{active} \in \{0,1\}$ $\mathsf{color} \in \{0,1\}$, $\mathsf{recruiting} \in \{0,1\}$, and $\mathsf{inEvalPhase} \in \{0,1\}$. The variable $\mathsf{Nbr} \in \{0,1\}^4$ stores the most recent message received, consisting of four

[6] More generally, we want $T = T_{\mathsf{inner}} \cdot \frac{1}{2}\log N$ for any $T_{\mathsf{inner}} = \omega(\log N)$. The first and last subphase will each be shorter by one round to account for the leader selection and evaluation phases.

[7] Recall that an agent's *neighbor* is the other agent the agent is randomly matched with in this round and that matchings in each round are independent and uniformly random.

boolean values (Nbr.inEvalPhase, Nbr.active, Nbr.color, Nbr.recruiting). An additional variable, to_recruit $\in [0, \frac{1}{2}\log N]$, is not necessary for the protocol itself but is used in the analysis. We emphasize that these variables are local to a single agent, so that different agents have independent copies of each of these variables. Initially, at the onset of the system, for each agent we will have that all variables are set to zero.

The variable round keeps track of which round it is within the epoch. The variable is incremented modulo T after each round, ensuring that agents begin and end each phase and each epoch at the same time.

The variables active and color specify whether an agent has been activated and colored, as well as the color of the agent. In the first round of each epoch, some of the agents will designate themselves as leaders and will become active, choosing at random a color color $\overset{\$}{\leftarrow} \{0, 1\}$, while the rest of the agents remain inactive. During the recruitment phase, additional agents will become active and will receive colors in $\{0, 1\}$, as inherited from a leader. The value of variable color is only relevant for active agents.

The variable recruiting specifies whether or not an active agent is trying to recruit in the present subphase. Each active agent should recruit only one additional agent in a single subphase, so this variable specifies whether or not the agent is still looking for an inactive agent to recruit in the subphase.

The variable inEvalPhase specifies whether the agent is currently in the evaluation phase, which is true exactly when round $= T - 1$.

Finally, the variable to_recruit specifies the number of additional followers an active agent is tasked with recruiting directly, which is the logarithm of the total number of agents that should be activated as a result of the given agent. When a new leader first becomes active, it sets to_recruit $= \frac{1}{2}\log N$, indicating that it is tasked with recruiting a total of $2^{\text{to\_recruit}} = \sqrt{N}$ agents. Each time a new agent is recruited, the value of to_recruit is decremented and shared with the newly recruited agent, indicated that each of the two is responsible for recruiting only half of the total. For instance, after the first time a leader recruits another agent, both agents will have to_recruit $= \left(\frac{1}{2}\log N\right) - 1$, and so each of the two agents subsequently are responsible for recruiting only $\sqrt{N}/2$ agents. In this way a leader can induce the recruitment of $\sqrt{N}$ agents in roughly a logarithmic number of rounds. Although the variable is not used by the algorithm itself, we will refer to it in the analysis.

We first present the main procedure run by each agent in every round. Each agent first exchanges messages with its neighboring agent in this round, if any. The agent then performs a consistency check on its on state and the state of its neighbor. Then, depending on the value of variable round modulo T, the program calls the appropriate subroutine for the corresponding phase.

Algorithm 1 Main protocol

```
 1: procedure MainProtocolStep
 2:     ExchangeMessages()
 3:     CheckRoundConsistency()

 4:     if round = 0 then                          ▷ Handle initialization of leaders
 5:         DetermineIfLeader()
 6:         round := round + 1
 7:     else if round < $T - 1$ then                       ▷ Perform recruitment phase
 8:         RecruitmentPhase()
 9:         round := round + 1
10:     else                          ▷ Final round of phase. Perform split or death.
11:         EvaluationPhase()
12:         round := 0
13:     end if
14: end procedure
```

We now describe the subroutine that exchanges messages with the neighboring agent, if any. An agent simply computes the indicator value of whether it is in the evaluation phase, and sends this information along with its activation state, color, and recruiting status. It receives a corresponding message from its neighbor, or the value $\perp$ if it has no neighbor in this round.

Algorithm 2 Subroutine to send and receive messages with neighboring agent

```
1: procedure ExchangeMessages
2:     if round $= T - 1$ then
3:         inEvalPhase $:= 1$
4:     else
5:         inEvalPhase $:= 0$
6:     end if
7:     MyStatus $:=$ (inEvalPhase, active, color, recruiting)
8:     Nbr $:=$ Communicate(MyStatus)
9: end procedure
```

We now describe the leader selection subroutine, which comprises the first round of each epoch. This is an entirely non-interactive process in which each agent becomes a leader with some fixed probability $1/(8\sqrt{N})$ by tossing its own coins, entirely independently of each other agent. With overwhelming probability, if the total number of agents is $\Theta(N)$, the number of leaders chosen in this phase will be $\Theta(\sqrt{N})$. Each newly activated leader chooses a random color in $\{0, 1\}$ and is tasked with recruiting $\sqrt{N}$ agents and assigning them this color.

Algorithm 3 Leader selection phase subroutine

```
1: procedure DetermineIfLeader
2:     active $:=$ TossBiasedCoin($\log(8\sqrt{N})$).
                         ▷ active $= 1$ with probability $1/(8\sqrt{N})$, and otherwise active $= 0$.
3:     if active $= 1$ then
4:         color $\xleftarrow{\$} \{0, 1\}$
5:         recruiting $:= 1$
6:         to_recruit $:= \frac{1}{2}\log(N)$
7:     end if
8: end procedure
```

The leader selection phase, as well as the evaluation phase below, requires the ability to flip biased coins with bias $1/\mathsf{poly}(N)$, in particular $1/\Theta(\sqrt{N})$, where bias refers to the probability of the coin having value 1 (so that a fair coin has bias 1/2). Recall that we assume only the ability to toss unbiased coins. We now give a simple procedure to obtain the desired bias using only $O(\log\log(N))$ bits of memory, assuming that $\log N$ is an even integer. More generally, we show how to obtain bias 2^{-a} for any integer a using $1 + \lceil \log a \rceil$ bits of memory. We note that it is sufficient to toss a coins and report 1 if they all landed heads and 1 otherwise. This requires counting to a, which can be done with $\log a$ memory.

Algorithm 4 Subroutine to toss a biased coin that equals 1 with probability 2^{-a} and equals 0 otherwise

```
1: procedure TossBiasedCoin(a)        ▷ Flip a biased coin with bias $\Pr[c = 1] = 2^{-a}$
2:     $c := 1$
3:     for $i = 1$ to $a$ do
4:         $b \xleftarrow{\$} \{0,1\}$
5:         if $b = 0$ then
6:             $c := 0$
7:         end if
8:     end for
9:     return $c$
10: end procedure
```

We now describe the recruitment phase, which is the second phase executed by each agent in every epoch and is the main source of interaction in the protocol. The phase lasts for $T = \Theta(\log^3 N)$ rounds, consisting of $\frac{1}{2}\log N$ subphases each of length $T_{\mathsf{inner}} = \Theta(\log^2 N)$ rounds. As discussed above, during this phase each leader is tasked with finding $\sqrt{N}$ inactive agents (i.e. with $\mathsf{active} = 0$) and coloring each of them with the color of the leader. We note again that the leader will not directly meet each of these $\sqrt{N}$ inactive agents, but rather that this is done by propagation, where the leader will activate some agents and each of these will activate additional agents. In each subphase each active agent will attempt to recruit a single nonactive agent, which will then start to recruit in the following subphase. Since there are $\frac{1}{2}\log N = \log\sqrt{N}$ subphases, if each attempt to recruit is successful, then a single leader will result in the activation of a total of $\sqrt{N}$ agents.

Algorithm 5 Recruitment phase subroutine

```
1: procedure RecruitmentPhase
2:     if Nbr.active = 0 and recruiting = 1 then          ▷ Other agent has been activated
3:         recruiting := 0
4:         to_recruit := to_recruit − 1
5:     else if active = 0 and Nbr.recruiting = 1 then     ▷ This agent must be activated
6:         active := 1
7:         color := Nbr.color
8:         recruiting := 0
9:         to_recruit := $\frac{1}{2}\log N - \lceil(\mathsf{round}+1)/T_{\mathsf{inner}}\rceil$
10:    end if
11:    if round $\equiv -1 \pmod{T_{\mathsf{inner}}}$ then                   ▷ Prepare for next round
12:        recruiting := 1.
13:    end if
14: end procedure
```

The final phase of the algorithm is the evaluation phase, which occurs on the last round of each epoch. In this phase, each matched active agent compares its color to that of its neighbor and makes a decision of whether to replicate itself or to self-destruct.

Algorithm 6 Subroutine to perform splitting and self-destruction at the end of each phase

```
1: procedure EvaluationPhase
2:     if active = 1 and Nbr.active = 1 then
3:         if Nbr.color = color then          ▷ Colors same: split with probability $1 - 16/\sqrt{N}$
4:             $c$ := TossBiasedCoin($\log(\sqrt{N}/16)$)
5:             if $c = 0$ then
6:                 Split()
7:             end if
8:         else                               ▷ Colors different: self-destruct with probability 1
9:             Die()
10:        end if
11:    end if
12:    active := 0
13:    color := 0
14:    recruiting := 0
15: end procedure
```

Finally, we give the subroutine invoked at the very beginning of each round, that performs a consistency check on the round values of the agent and its neighbor. In the absence of adversarial insertions this subroutine is unnecessary, since agents will always have the correct round value in the epoch. However, it is necessary if the adversary is allowed to insert agents with an incorrect round value. If left to increase unchecked, the presence of many agents with different round values would interfere with the operation of the protocol. We will prevent this from happening by causing agents to self-destruct as soon as they encounter an agent with a different round value. However, implementing this exactly would require $\Theta(\log\log N)$-bit messages, since agents would need to exchange their round numbers. To avoid this, we will instead have agents exchange only an indicator variable for whether or not they are in the evaluation phase. An agent will self-destruct if it is in the evaluation phase and its neighbor is not, or if its neighbor is in the evaluation phase and it is not. This process deletes a small number of agents with the correct round number along with agents with the incorrect round number. We will show in the analysis below that this will ensure that there are few agents with the incorrect round number and that only a small number of agents with the correct round number will self-destruct as a result of this procedure.

Algorithm 7 Subroutine to determine if agent knows the correct round

```
1: procedure CheckRoundConsistency
2:     if $\mathsf{Nbr} \neq \perp$ and $\mathsf{inEvalPhase} \neq \mathsf{Nbr.inEvalPhase}$ then
3:         Die()
4:     end if
5: end procedure
```

2.4 Analysis

In this section we prove the main theorem.

Theorem 2.4.1. *Let $\alpha, \gamma, \varepsilon$ be positive constants, where $\gamma \leq 1$ is a lower bound on the fraction of agents that is matched in each round. Then Algorithm 1 is a population stability protocol using $\omega(\log^2 N)$ states per agent and three-bit messages*[8] *guaranteeing that if the adversary inserts and deletes at most $K = O(N^{1/4-\varepsilon})$ agents in each round, then with all but negligible probability the population will remain between $(1-\alpha)N$ and $(1+\alpha)N$ for any polynomial number of rounds.*

For ease of presentation, the version of the protocol described above uses four-bit messages. However, we can reduce the message size to three bits, as follows. If the agent is in the evaluation phase (i.e. $\mathsf{inEvalPhase} = 1$) then the message must contain the values active and color, but need not contain $\mathsf{recruiting}$. If $\mathsf{inEvalPhase} = 0$, then the message must contain the value color but not active if $\mathsf{recruiting} = 1$ and the value active but not color if $\mathsf{recruiting} = 0$. Consequently, the desired information can be encoded in only three bits.

[8]A straightforward implementation of the protocol described above would use $\Theta(\log^4 N)$ states and four-bit messages. We describe below how to achieve the improved bounds stated here.

For the memory requirements, $\log T$ bits are needed to store the variable $\textsf{round} \in \{0,1\}$, and the other variables stored by each agent consist of eight boolean values. The invocations of the subroutine TossBiasedCoin() require $\log\log N$ bits of local memory. However, the subroutine is only invoked in two rounds of each epoch, the leader selection round and the evaluation round. Consequently, using additional indicator bits to specify whether an agent is in each of these those two rounds, the memory used to store the variable **round** can be used as the helper memory for the subroutine, and so additional memory is not necessary. For $T = \frac{1}{2}\log^3 N$, the total number of states is therefore $\Theta(\log^3 N)$. However, it suffices to have $T = T_{\textsf{inner}} \cdot \log N$ for any $T_{\textsf{inner}} = \omega(\log N)$, and so we can reduce the number of states to $\omega(\log^2 N)$.

Roadmap to proof We must show that the population size will remain close to the target value N. We will do this by means of two key steps.

The first step is to show that in any single epoch the population size will be relatively stable. That is, we show that the population in the middle or end of an epoch will not be too much larger or smaller than the population at the beginning of the epoch. We achieve this by showing that irrespective of the adversary's actions, with overwhelming probability the number of agents of each color in the evaluation phase will be concentrated around one-sixteenth of the total number of agents. This step is formalized in Lemmas 2.4.5 and 2.4.6 in Section 2.4.2.

The second step is to show that the population size will tend to correct itself if it has deviated too far from the target value N. More precisely, we show that if the population is far from N, then in expectation the population at the end of the phase will be substantially closer to N than the population at the start of the phase. This step highlights a key tension of the proof, namely the difficulty analyzing a system that contains both random components in the matching schedule and worst-case components in the adversary's insertions and deletions. Indeed, it is not even clear a priori what it means to discuss expectation in a system with a worst-case adversary. This step is formalized in Lemma 2.4.7 in Section 2.4.3.

Putting these two steps together will enable us to conclude the proof of the theorem.

Consider the first epoch in which the population size lies outside the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$. Since the population size does not deviate too much in a single epoch, the population at the start of the next epoch will be close to $(1 \pm \frac{\alpha}{2})N$. But for each epoch in which the population remains outside the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$, in expectation the change in population will be in the direction of the target value N. Considering the next $N^{0.01}$ epochs, a Chernoff-Hoeffding bound then implies that with overwhelming probability the population will return to the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$. Since the population does not change much in each epoch, it follows further that the population will remain inside the interval $[(1-\alpha)N, (1+\alpha)N]$ during these $N^{0.01}$ epochs. We will conclude that with all but negligible probability, the population will remain in the interval $[(1-\alpha)N, (1+\alpha)N]$ for any polynomial number of rounds.

We now outline the remainder of the section. In Section 2.4.1 we prove some preliminary lemmas about the protocol. These lemmas provide us with invariants that we will need in order to prove the two key steps above. In particular, we show that nearly every agent knows the correct round in the epoch, that at least half of the agents are inactive (i.e. have $\mathsf{active} = 0$) at any point in the execution of the protocol, and that any leader selected in the first round of the epoch will succeed in recruiting a full cluster of size $\sqrt{N}$ unless either the adversary deletes some agent in that cluster or an agent with the wrong round number interferes with the recruitment.

In Section 2.4.2 we prove the first of the key steps, showing that with high probability the population size does not deviate too much in a single epoch. In Section 2.4.3 we prove the second of the key steps, showing that if the population has drifted too far from the target value, then in expectation the population will correct itself. Finally, in Section 2.4.4 we conclude the proof of the main theorem.

2.4.1 Bookkeeping lemmas

We will first prove several bookkeeping lemmas to guarantee that with overwhelming probability, certain invariants continue to hold throughout the execution of the protocol. We will subsequently use these invariants to prove stronger statements that will enable us to conclude the correctness the protocol.

Recall that during the protocol, agents keep track of the current round within the epoch, that is, the round number modulo $T = \log^3 N$. However, the adversary is empowered to insert new agents into the system with arbitrary initial state, and in particular may insert agents with the incorrect round number. Our first lemma provides a bound on the number of agents with the incorrect round number. Recall that $\gamma = \Theta(1)$ is a lower bound on the fraction of parties in each round that are matched. We can assume that $\alpha \leq 1/2$, since the desired statement is stronger for smaller α.

Lemma 2.4.2. *For any $t > 0$, suppose that the population size remains above $N_{\mathsf{min}} = N/2$ for the first t rounds of the protocol. Then there exists some negligible function ν such that conditioning on this event, with probability $1 - t \cdot \nu(N)$, all but $(1 + \gamma^{-1})N^{1/4}$ of the agents will have the same value for variable* round *in each of these t rounds.*

Proof. We prove this by induction on the round number. Initially all agents have round $= 0$. Assume for induction that at round r all but $\gamma^{-1}N^{1/4}$ of the agents have variable round $= 0$. We will show that with high probability, the same statement holds at the start of round $r + T$. The adversary may add an additional K agents in each of these T rounds, for a total of $KT \leq N^{1/4}/8$ agents. Note that each agent with the wrong round value may split at most once in this epoch of T rounds, when it reaches its evaluation phase. If there are $v = O(N^{1/4})$ agents which differ from the majority value for variable round, then the probability of such an agent being matched with another such agent in its evaluation phase is at most $\gamma \cdot \frac{v}{N_{\mathsf{min}}} = \gamma \cdot O(N^{-3/4})$. It follows that with high probability, the number of agents with the wrong round value that split during this epoch is at most $N^{0.01}/8 < N^{1/4}/8$. Consequently at any point in this epoch of T rounds, the number of agents with the wrong round value will be at most $\gamma^{-1}N^{1/4} + N^{1/4}/4$. Each agent with the wrong round value at the start of the majority evaluation phase (i.e. round $r + T - 1$) has probability at least $\gamma(N_{\mathsf{min}} - (\gamma^{-1} + 1/4)N^{1/4})/(N_{\mathsf{min}}) \geq \gamma - 3N^{-3/4}$ of being matched with an agent starting the evaluation phase, so with all-but-negligible probability, at most $(1/\gamma - 1/2)N^{1/4}$ of these agents will not be matched with an agent with different round value and will survive the round. It follows that at the start of round $r+T$, at most $\gamma^{-1}N^{1/4}$ agents have round value different from 0. Consequently, by induction we have that with overwhelming probability, the number of agents with variable round $\neq 0$ in any round $r \equiv 0 \pmod{T}$ is at most $\gamma^{-1}N^{1/4}$. Since we showed above that the number of agents with the wrong round number that can be added during the epoch is small with overwhelming probability, the lemma follows. □

Lemma 2.4.3. *With high probability, if the population is in the interval* $[(1-\alpha)N,(1+\alpha)N]$ *at the start of an epoch, at any point in the epoch, at most* $1/2$ *of the agents have* active $= 1$.

Proof. Let $m \in [(1-\alpha)N,(1+\alpha)N]$ be the population at the start of the epoch. With all but negligible probability the number of leaders chosen will be $m/(8\sqrt{N}) \pm o(N^{0.26})$ by a Chernoff bound. Each leader may induce the activation of at most $\sqrt{N}$ total agents. During the epoch, the adversary may insert an additional $T\cdot K \leq N^{1/4}$ agents, which may each induce the activation of $\sqrt{N}$ total agents. Consequently at any point in the epoch, the number of active agents will be at most $m/8+o(N^{0.76})$. Prior to the evaluation step, the adversary can have killed at most $T\cdot K \leq N^{1/4}$ agents. By the previous lemma, at most $\gamma^{-1}N^{1/4}$ agents at the start of the epoch can have the wrong value for round, and at most $T\cdot K$ additional such agents can be introduced during the protocol, so at most $O(\gamma^{-1}N^{1/4}) = O(N^{1/4})$ agents can be killed in the CheckRoundConsistency() procedure. Consequently the population throughout the epoch until the evaluation phase will be at least $m-O(N^{1/4})$. The conclusion follows. □

Lemma 2.4.4. *Suppose the population is in the interval* $[(1-\alpha)N,(1+\alpha)N]$ *at the start of an epoch. Then with high probability, in the last round of the epoch, every active agent entering the evaluation phase that was not inserted by the adversary during this epoch will have* to_recruit $= 0$.

Proof. By Lemma 2.4.3, at most half of the agents are active in each round, so the probability in each round of encountering an inactive agent at each round is at least $\gamma/2 = \Theta(1)$. With overwhelming probability, in any sequence of $\omega(\log N)$ steps an agent will encounter an inactive agent and will be able to recruit it. Applying a union bound, we have that in each cycle of T_{inner} steps, each of the $O(N)$ active agents attempting to recruit will be successful in finding an inactive agent to recruit. Consequently each agent will be able to recruit the desired number of additional agents, and the lemma follows. □

2.4.2 Bounded deviation

In this section we show that with high probability, the population size does not change by too much in any single epoch.

Lemma 2.4.5. *Let m be the population at the start of an epoch. For any constant $\delta > 0$, with high probability, the number of agents with each color $b \in \{0, 1\}$ at the start of the evaluation phase will be $m/16 \pm o(N^{3/4+\delta})$.*

Proof. Let m be the population at the start of the epoch. With all but negligible probability, the number of leaders selected with color 0 at the beginning of the epoch will be $m/(16\sqrt{N}) \pm o(N^{1/4+\delta})$, and similarly for the number of leaders selected with color 1. In the absence of adversarial deletions, each leader will recruit $\sqrt{N}$ followers with the same coin value, inducing the presence of $m/16 \pm o(N^{3/4+\delta})$ agents of each color by the final round of the epoch.

The adversary may insert or delete $K \cdot T = \tilde{O}(N^{1/4-\varepsilon})$ agents over the course of the epoch. Each inserted agent can induce the activation of at most $\sqrt{N}$ additional agents, and similarly each removed agent could have activated up to $\sqrt{N}$ additional agents. Additionally, by Lemma 2.4.2, at most $O(N^{1/4})$ agents will be removed in procedure CheckRoundConsistency() upon encountering an agent with a different value for variable round. Overall the actions of the adversary can affect the number of agents of each color by at most $O(N^{3/4-\varepsilon})$. It follows that despite adversarial action, the number of agents of each color at the start of the evaluation phase will be $m/16 \pm o(N^{3/4+\delta})$ with all but negligible probability. □

Lemma 2.4.6. *For any $\delta > 0$, with all but negligible probability, if the population is in the interval $[(1-\alpha)N, (1+\alpha)N]$ at the start of an epoch, the population will have deviated by at most $O(N^{1/2+\delta})$ by the end of the epoch.*

Proof. Let $m \in [(1-\alpha)N, (1+\alpha)N]$ be the population at the start of the epoch. By the previous lemma, at the start of the evaluation phase the number of agents with color 0 will be $m_0 = m/16 \pm O(N^{3/4+\delta})$ with all but negligible probability, and likewise the number of agents with color 1 will be $m_1 = m/16 \pm O(N^{3/4+\delta})$. We condition on these events. Since the adversary can insert at most K agents in each of the $T = \log^3 N$ rounds for a total of $\tilde{O}(N^{1/4-\varepsilon})$, and no other new agents with the correct value of variable round can be produced until the evaluation phase, Lemma 2.4.2 implies that the total number of agents at the start of the evaluation phase is at most $m + O(\gamma^{-1}N^{1/4})$. Similarly, since the adversary can have directly removed at most $K \cdot T = K \log^3 N$ agents, and at most $2((\gamma^{-1}+1)N^{1/4} + K\log^3 N)$ agents with the correct value of round may have been removed as a result of procedure CheckRoundConsistency() after encountering an agent with a different value of round, it follows that for $\gamma = \Theta(1)$, the total number of agents at the start of the evaluation phase is $m \pm O(N^{1/4})$.

Consequently the communication graph for the evaluation phase is a random matching of size $q = \Omega(\gamma N) = \Omega(N)$. Sample such a matching by first choosing a set of q left vertices and a set of q right vertices, and then associating corresponding vertices on the left and right. Let $q_0 = qm_0/m'$ and $q_1 = qm_0/m'$. With high probability the number of left (respectively, right) vertices with color 0 will be $q_0 \pm \tilde{O}(\sqrt{N})$, and similarly for vertices of color 1.

It follows that with all but negligible probability, the number of left-vertices of color b that split after being matched with a right-vertex of color b is $q_b^2/q \pm \tilde{O}(\sqrt{N})$ for each $b \in \{0,1\}$. Similarly, the number of left-vertices of color b that self-destruct after being matched with a right-vertex of color $1-b$ is $q_b q_{1-b}/q \pm \tilde{O}(\sqrt{N})$. Consequently will all but negligible probability, noting that $q_0 - q_1 = O(N^{3/4+\delta})$, we have that the change in population during the evaluation phase is

$$\frac{2}{q} \cdot (q_0^2 + q_1^2 - 2q_0q_1) \pm \tilde{O}(\sqrt{N}) = \tilde{O}(N^{1/2+2\delta})$$

yielding the desired result since δ is an arbitrary positive constant.

□

2.4.3 Correcting population drift

In this section we show that if the population has drifted too far from the target value N, in expectation it will tend to correct itself.

Lemma 2.4.7. *If the population is in the interval $[(1-\alpha)N, (1-\frac{\alpha}{2})N]$ at the start of an epoch, then for any adversarial strategy, in expectation the population will increase by $\Omega(\sqrt{N})$ by the end of the epoch. If the population is in the interval $[(1+\frac{\alpha}{2})N, (1+\alpha)N]$ at the start of an epoch, then in expectation the population will decrease by at least $\Omega(\sqrt{N})$ by the end of the epoch.*

Proof. The behavior of the system during the evaluation phase depends on the distribution of coin values of active agents in this phase. We would like to argue that each pair of clusters has its colors assigned by independent, fair coin flips. This is clearly true in the absence of an adversary. However, in our setting, adversarial instertions and deletions can bias the joint distribution of the colors of a pair of agents in different clusters.[9] Nonetheless, we will argue that the adversary's influence is limited, and that for most pairs of agents in different clusters, we can think of the joint distribution of colors as unbiased and uniform. We will label clusters as *honest* or *adversarial*, where the colors of a pair of honest clusters can be regarded as independently sampled, and no assumption is made on the colors of the adversarial clusters.

Consider any fixed adversarial strategy. Note that the adversary can insert or delete a total of no more than $2 \cdot T \cdot K = \tilde{O}(N^{1/4-\varepsilon})$ agents during the epoch, and consequently can influence no more than this many clusters. Any strategy of the adversary during this epoch can be emulated by deferring any deletions of colored agents (with the correct round value) to the beginning of the evaluation phase, deleting instead an inactive agent. Recall that by Lemma 2.4.4, each cluster not affected by the adversary will consist of $\sqrt{N}$ agents at the beginning of the evaluation phase. At the beginning of the evaluation phase, we allow this

[9]For instance, in the first round of the epoch, the adversary can instert additional leaders all with color 0, or can delete several leaders that have color 1. This difficulty arises because we allow the adversary to observe the internal memory of all agents, including the results of coin tosses.

new adversary not only to delete the specified agent, but also to set $\mathsf{active} = 0$ for any subset of the $\sqrt{N} - 1$ other agents in the cluster. Note that any attack that could be accomplished by the original adversary can still be carried out by this new adversary that defers all of its deletions of colored agents to the evaluation phase but is subsequently allowed to modify up to $\tilde{O}(N^{3/4-\varepsilon})$ agents in $\tilde{O}(N^{1/4-\varepsilon})$ different clusters.

Since we now defer deletions of colored agents to the beginning of the evaluation phase, each cluster induced by a leader selected in the first round of the epoch will have the full $\sqrt{N}$ members, and consequently these clusters are indistinguishable except for their color. Consequently as long as the adversary can modify agents in $\tilde{O}(N^{1/4-\varepsilon})$ clusters of each color, *which* specific cluster of each color is irrelevant. Consider an arbitrary indexing of the agents before the first round of the epoch, and consider the first $\tilde{O}(N^{1/4-\frac{1}{2}\varepsilon})$ leaders chosen in this round. With all but negligible probability, this set will contain at least $\tilde{O}(N^{1/4-\varepsilon})$ agents of each color $\mathsf{color} \in \{0, 1\}$, and so the strategy of the adversary can be carried out by manipulating only the clusters of agents in this set. Let the clusters induced by these agents be the *adversarial clusters* along with the clusters induced by any agents inserted by the adversary, and let the remaining clusters be the *honest clusters.*

Now we have reduced to a setting in which we have achieved the desired property that each honest cluster has size $\sqrt{N}$, and the coin flips of any two honest clusters are independent. However, we are now dealing with a modified adversary that can affect a larger overall number of agents. Let m' be the number of agents at the start of the evaluation phase. By Lemma 2.4.5, it follows that with all but negligible probability, the number of agents in honest clusters is $m_h = m'/8 \pm o(N^{3/4+\delta})$, and the number of agents in clusters influenced by the adversary is $m_a = O(N^{3/4-\frac{1}{2}\varepsilon})$.

Pick a random pair of vertices at the start of the evaluation phase. Then with probability $1/64 - O(N^{3/4+\delta}/m') = \Omega(1)$ both agents will belong to honest clusters, with probability $O(N^{3/4-\frac{1}{2}\varepsilon}/m')$ one agent will belong to an honest cluster and the other to an adversarial cluster, and with probability $o(N^{3/2}/m'^2)$ both will belong to adversarial clusters.

A pair of vertices belonging to honest clusters will have the same color if they belong to the same cluster, and independently random colors if they belong to different clusters. Consequently the probability that such a pair of vertices will have the same color is $\frac{1}{2} + \frac{\sqrt{N}}{2m_h} - O(\frac{1}{m_h})$. It follows that the expected change in population resulting from the matching of a pair of vertices belonging to honest clusters is

$$\left(1+\frac{\sqrt{N}}{m_h}\right)\cdot\left(1-\frac{16}{\sqrt{N}}\right)-\left(1-\frac{\sqrt{N}}{m_h}\right)-O\left(\frac{1}{m_h}\right)=\frac{2\sqrt{N}}{m_h}-\frac{16}{\sqrt{N}}-\frac{O(1)}{m_h}.$$

Recalling that $\alpha \leq 1/2$ is a fixed constant, for $m < (1-\frac{\alpha}{2})N$ this quantity is $\Theta(1/\sqrt{N})$, and for $m > (1+\frac{\alpha}{2})N$ this quantity is negative, with magnitude $\Theta(1/\sqrt{N})$.

The honest clusters consist of nearly the same number of agents of each color ($m/16 \pm o(N^{3/4+\delta})$ of each). It follows that the expected change in population resulting from the matching of a vertex in an honest cluster and a vertex in an adversarial cluster has magnitude $o(N^{-1/4+\delta})$. Making no assumption about the distribution of colors in the adversarial clusters, the matching of two vertices in an adversarial cluster can have a change in population of $O(1)$.

Consider a random pair of vertices at the start of the evaluation phase. For $m < (1-\frac{\alpha}{2})N$, the expected change in population resulting from matching this pair of vertices is $\Omega(N^{-1/2}) - o(N^{-1/4+\delta}N^{-1/4-\frac{1}{2}\varepsilon}) - o(N^{-1/2}\cdot 1) = \Omega(N^{-1/2})$, where we choose $\delta \leq \varepsilon/2$. Since the number of matched pairs of vertices in the evaluation phase is $\gamma m' = \Theta(N)$, it follows from linearity of expectation that the expected change in population during the evaluation phase is $\Omega(\sqrt{N})$. Similarly, for $m > (1+\frac{\alpha}{2})N$ we have that the expected change in population during the evaluation phase is $-\Omega(\sqrt{N})$. The adversary can delete or insert only $K \cdot T = o(N^{1/4})$ agents during the epoch, and Lemma 2.4.2 implies that $O(N^{1/4})$ agents will self-destruct during procedure CHECKROUNDCONSISTENCY() during the epoch, so the other terms dominate, and the conclusion follows. □

2.4.4 Putting everything together

We now show that if the population size leaves the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$ during an epoch, with high probability it will return to this interval during one of the next few epochs. With this final lemma, we then conclude the proof of the theorem.

Lemma 2.4.8. *Consider an epoch in which the population has drifted outside the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$. With all but negligible probability, the population will once again be in the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$ at the start of one of the next $t = N^{0.01}$ epochs.*

Proof. Assume to the contrary, and let epoch 0 denote the first epoch after the population has exceeded the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$. For concreteness, suppose the population has dropped below $(1-\frac{\alpha}{2})N$. By Lemma 2.4.6, with all but negligible probability the population is still above $(1-\frac{\alpha}{2})N - O(N^{0.501})$. For $i \in \{1,\ldots,t\}$, let X_i be the random variable denoting the difference in population between the start of epoch i and the start of epoch $i-1$, and let $\overline{X} = (X_1 + \cdots + X_t)/t$. By Lemma 2.4.6, with all but negligible probability, each random variable X_i is bounded in the range $[-O(N^{0.501}), O(N^{0.501})]$. Since by assumption the population is below $(1-\frac{\alpha}{2})N$ at the start of each epoch, Lemma 2.4.7 implies that $\mathbb{E}[X_i] = \Omega(\sqrt{N})$. It follows by a Chernoff-Hoeffding bound that with all but negligible probability, for any constant c, $\left|\overline{X} - \mathbb{E}[\overline{X}]\right| \leq c\sqrt{N}$, so with all but negligible probability we have that $\overline{X} = \Omega(\sqrt{N})$ and $X_1 + \cdots + X_t = \Omega(N^{0.51})$. Consequently the population at the start of epoch t exceeds $(1-\frac{\alpha}{2})N$. The argument is identical when the population has exceeded $(1+\frac{\alpha}{2})N$. □

We now conclude the proof of Theorem 2.4.1.

Proof of Theorem 2.4.1. Consider any polynomial f, and suppose that for some adversarial strategy the population deviates from the interval $[(1-\alpha)N, (1+\alpha)N]$ in $f(N)$ rounds with non-negligible probability. It follows that for some pair of epochs $i < j \in [1 + \lfloor f(N)/T \rfloor]$, with non-negligible probability the population deviates from interval $[(1-\alpha)N, (1+\alpha)N]$ for the first time in epoch j after deviating from the interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$ for the last time in epoch i. We condition on this event. Lemma 2.4.6 implies that until epoch j, with all but neligible probability the population will deviate in each epoch by at most $\tilde{O}(\sqrt{N})$. But then Lemma 2.4.8 implies that with all but negligible probability the population will return to interval $[(1-\frac{\alpha}{2})N, (1+\frac{\alpha}{2})N]$ within $N^{0.01}$ epochs, which is a contradiction. Consequently the population will remain between $(1-\alpha)N$ and $(1+\alpha)N$ with high probability for any polynomial number of rounds, as desired. $\square$

Part II

Cryptography

Chapter 3

Network oblivious transfer

This chapter is based on joint work with Ranjit Kumaresan and Srinivasan Raghuraman [KRS16].

3.1 Introduction

Protocols for secure multiparty computation [Yao86, GMW87, BGW88, CCD88] allow a set of mutually distrusting parties to carry out a distributed computation without compromising the privacy of inputs or the correctness of the end result. As a research area, secure computation has witnessed several breakthroughs in the last decade [ZRE15, KS08, IKNP03, NNOB12, LP07, Lin13, HKE13, MR13, HKK$^+$14, LR14]. However, despite a wide array of potential game-changing applications, there is little practical adoption of secure computation today (with the notable exceptions of [BLW08, BCD$^+$09]). Computations wrapped in a secure computation protocol do not yet deliver results efficiently enough to be acceptable in many cloud-computing applications. For instance, state-of-the-art semihonest 2-party protocols incur a factor ≈ 100 slowdown even for simple computations.

In the absence of practical real-world protocols for secure computation which are secure in the presence of any number of dishonest parties, there is a need for relaxations that are meaningful and yet provide significant performance benefits. As an example, classic protocols for secure computation [BGW88, CCD88, RB89] (with subsequent improvements e.g., [DI06, BSFO12, BH08, DN07a, DIK$^+$08, DIK10]) offer vastly better efficiency at the cost of tolerating only a small constant fraction of adversaries. The resilience offered is certainly acceptable when the number of participating parties is large, e.g., the setting of *large-scale* secure computation [BCP15, DKMS12, ZMS14, BGT13]. Although large-scale secure computation is well-suited for interesting applications such as voting, census, and surveys, many natural settings involve computations over data supplied by a few end users. In such cases, the overhead associated with interaction among a large number of *helper parties* is likely to render these protocols more expensive than a standard secure computation protocol among the end users. If the number of helper parties is small, security against a small fraction of corrupt parties may be a very weak guarantee, since a handful of corrupt parties could render the protocol insecure.

An orthogonal approach for reducing the online cost of secure computation protocols is the use of *preprocessing* [Bea95, DPSZ12, BDOZ11, AIKW13]. This approach can dramatically reduce the cost of secure computation: for instance, given preprocessing [Bea95], the ≈ 100 factor slowdown for simple computations no longer applies. Recent theoretical research has shown that many primitives can even be made *reusable* (e.g. [GKP$^+$13]). Perhaps the most important drawback of this approach (other than the fact that the preprocessing phase is typically very expensive) is that the preprocessing is not *transferable*. Clearly, a pair of parties that want to perform a secure computation cannot benefit from this approach without performing the expensive preprocessing step. Moreover, this seems to hold even if each of the two parties have set up the preprocessing with multiple others. Typically, the cost of the preprocessing phase is quite high, presenting a barrier for the practical use of preprocessed protocols. This is especially true in settings where parties are unlikely to run many secure computations that would amortize the cost of preprocessing.

Motivated by the discussion above, we conclude that some directions that seem to offer efficiency benefits for secure computation are (1) highly resiliant protocols that use only a small number of helper parties, and (2) a preprocessing procedure that allows a notion of transferability between users. Taken together, these two ideas have the potential to provide an *infrastructure* for efficient secure computation. Some sets of parties might run a preprocessing phase among themselves. These parties can then act as helper parties and "transfer" their preprocessing to help users who want to run a secure computation protocol. We informally describe some desiderata for such an infrastructure:

- *Reusability/Amortization.* Setting up an infrastructure component could be expensive, but using it and maintaining it should be inexpensive relative to setting up a new component.

- *Transferability/Routing.* It should be possible to combine different components of the infrastructure to deliver benefits to the end users.

- *Robustness/Fault-tolerance.* Failure or unavailability of some components of the infrastructure should not nullify the usefulness of the infrastructure.

It is not hard to see that the above criteria are fulfilled for infrastructures that we use in daily life, for instance the infrastructure for online communication consisting of undersea transatlantic cables, routers, wireless access points, and so forth. What cryptographic primitives would be good candidates for a *secure computation infrastructure*? In this work, we explore the possibility of using *oblivious transfer* [Rab81, EGL82] for this purpose.

3.1.1 Our Model: Network Oblivious Transfer

Oblivious transfer (OT) is a fundamental building block of secure computation [Kil88, IPS08]. As discussed in [IPS08], some of the benefits of basing secure computation on OT include:

- *Preprocessing.* OT enables precomputation in an offline stage before the inputs or the function to be computed are known. The subsequent online phase is extremely efficient [Bea95].

- *Amortization.* The cost of computing OTs can be accelerated using efficient OT extension techniques [Bea96, IKNP03, IPS08, NNOB12].

- *Security.* OTs can be realized under a wide variety of computational assumptions [PVW08, EGL82, Rab81, NP01, CO15] or under physical assumptions.

In this work, we consider n parties connected by a synchronous network with secure point-to-point private communication channels between every pair of parties. In addition, some pairs of parties on the network have established *OT channels* between them providing them with the ability to perform arbitrarily many OT operations. We represent the OT channel network via an *OT graph* G. The vertices of G represent the n parties, and pairs of parties that have an established OT channel are connected by an edge in G. Since OT can be reversed unconditionally [WW06], we make no distinction between the sender and the receiver in an OT channel. This OT graph represents the infrastructure we begin with. The OT channels could either represent $\mathsf{poly}(\kappa)$ 1-out-of-2 OT correlations for a computational security parameter κ, or a physical channel (e.g., noisy channel) that realizes, say δ-Rabin OT [Rab81].[1] We are interested in obtaining security against adaptive semihonest adversaries. We also discuss security against adaptive malicious adversaries under computational assumptions.

Two parties that are connected by an edge can use the corresponding existing OT channel to run a secure computation protocol between themselves. What about parties that are not connected by an edge? Clearly, they can establish an OT channel between themselves via an OT protocol [PVW08, CO15] or perhaps using a physical channel. The latter option, if possible, is likely to be expensive and the costs of setting up a physical channel may be infeasible unless the two parties are likely to execute many secure computation protocols. The former option is also expensive as it involves use of public-key cryptography which is somewhat necessary in the light of [IR89].[2] This motivates the question of whether additional parties can use an existing OT infrastructure to establish an OT channel between themselves unconditionally or relying only on the existence of symmetric-key cryptography. A positive

[1]Recall that κ 1-out-of-2 OT correlations can be extended to $\mathsf{poly}(\kappa)$ 1-out-of-2 OT correlations via OT extension using just symmetric-key cryptography (e.g. one-way functions [Bea96] or correlation-robust hash functions [IKNP03]).

[2]As a rule of thumb, use of public-key cryptography is computationally around 4-6 orders of magnitude more expensive than using symmetric-key cryptography [BHKR13].

result to this question would show that expensive cryptographic operations are not required to set up additional OT channels which could be used for efficient secure computation. In this work we construct OT protocols with information-theoretic security against a threshold adversary.

The generality of an OT infrastructure. Consider the following candidate for an infrastructure. Suppose there is a channel between a pair of parties that allows them to securely evaluate any function. Since OT is complete for secure computation, one can apply the results of [IPS08, Kil88] to use the OT channel to implement a secure evaluation channel. In the other direction, one can use a secure evaluation channel to trivially implement OT channels. Consequently, such a channel is equivalent to an OT channel. The same argument extends to channels that implement any 2-party primitive that is complete for secure computation [MPR10, BMM99]. Furthermore, the above argument also applies to the setting where a *set* of parties have a secure evaluation channel. Such a channel is equivalent to an OT graph where parties in the set have pairwise OT channels with everyone in the set.

Assuming a full network of secure channels. Secure channels between two parties can be implemented either via non-interactive key exchange and hybrid encryption or via a physical assumption. We emphasize that the one-time setup cost of emulating a secure channel (e.g. via Diffie-Hellman key exchange) is much lower than the one-time setup cost of emulating an OT channel that allows unbounded OT calls via an OT protocol even using OT extension. Furthermore, our assumption of secure channels is identical to the setting of [Kil88, GV87, IPS08], who show that secure computation reduces to OT under information-theoretic reductions.

3.1.2 Related Work and Our Contributions

Related work. As mentioned previously, there is a large body of work on secure computation in the offline/online model (cf. [LPSY15, LOS14, DPSZ12, BDOZ11, NNOB12] and references therein). These protocols exhibit an extremely fast online phase at the expense of a slow preprocessing phase (sometimes using MPC [LPSY15] or more typically, OT correlations [NNOB12] or a somewhat homomorphic encryption scheme [DPSZ12]). To the best of our knowledge, the question of *transferability* of preprocessing has not been explicitly investigated in the literature with the notable exception of [HIK07], which we will discuss in greater detail below. There is a large body of work on secure computation against a threshold adversary (e.g. [BGW88, CCD88, RB89, GMW87]). Popular regimes where secure computation against threshold adversaries have been investigated are for $t < n/3$, $t < n/2$, or $t = n-1$. In this work we are interested in threshold adversaries for a dishonest majority, that is, adversaries which can corrupt t out of n parties for $n/2 \leq t < n$.[3] Such regimes were investigated in other contexts such as authenticated broadcast [GKKO07] and fairness in secure computation [BOO10, HLM13, IKLP06]. Infrastructures for *perfectly secure message transmission* (PSMT) were investigated in the seminal work of [DDWY90] (see also [FFGS07] and references therein). While the task of PSMT is similar to our question regarding OT channels, there are inherent differences. For example, our protocols can implement OT even between two parties that are isolated in the OT graph (i.e., not connected to any other party via an OT channel).[4] In PSMT, on the other hand, there is no hope of achieving secure communication with a node that is not connected by any secure channel.

Most relevant to our results is the work of Harnik, Ishai, and Kushilevitz [HIK07]. The main question in their work is an investigation of the number of OT channels sufficient to implement a n-party secure computation protocol. In a nutshell, they show against an adaptive t-threshold adversary for $t = (1-\delta)n$, an explicit construction of an OT graph consisting of $(n + o(n))\binom{\lceil 1/\delta \rceil}{2}$ OT channels that suffices to implement secure computation among the n parties. They note further that against a static adversary, $\binom{\lceil s/\delta \rceil}{2}$ OT channels suffice, where s denotes a statistical security parameter. On the negative side, they show that a complete OT graph is necessary for secure computation when dealing with an adversary that can corrupt $t = n-1$ parties. They derive this result by showing that in a 3-party OT graph with two OT channels, it is not possible to obtain OT correlations between the

[3]When $t < n/2$, there is no need to rely on an OT infrastructure [RB89].

[4]Recall that the model considered in this work, we assume a *full* network of secure private communication channels.

third pair of parties with security against two corruptions. Moreover they generalize their 3-party negative result to any OT graph whose complement contains the complete bipartite graph $K_{n-t,n-t}$ as a subgraph. In our paper we extend and generalize the results of [HIK07], fully characterizing the networks for which it is possible to obtain OT correlations between a designated pair of parties. We now proceed to explain our contributions in more detail.

Our contributions. We introduce our main result:

Theorem (informal). Let $G = (V, E)$ be an OT graph on n parties $P_1, \ldots P_n$, so that any pair of parties P_i, P_j which are connected by an edge may make an unbounded number of calls to an OT oracle. Let $\mathbb{A}$ be the class of semihonest t-threshold adversaries which may adaptively corrupt at most t parties.[5] Then two parties A and B in $\{P_1, \ldots, P_n\}$ can information-theoretically emulate an OT oracle while being secure against all adversaries $\textsc{adv} \in \mathbb{A}$ if and only if

1. (honest majority) it holds that $t < n/2$; or
2. (trivial) A and B are connected by an edge in G; or
3. (partition) there exists no partition V_1, V_2, V_3 of G such that all of the following conditions are satisfied: (a) $|V_1| = |V_2| = n - t$ and $|V_3| = 2t - n$; (b) $A \in V_1$ and $B \in V_2$; and (c) for every $A' \in V_1$ and $B' \in V_2$ it holds that $(A', B') \notin E$.

Our main theorem gives a complete characterization of networks for which a pair of parties can utilize the OT network infrastructure to execute a secure computation protocol. The first two conditions in our theorem are straightforward: (1) if $t < n/2$, then we are in the honest majority regime, and thus it is possible to implement secure computation (or emulate an OT oracle) using the honest majority information-theoretically secure protocols of [RB89]; (2) clearly if A and B are connected by an OT edge then by definition they can emulate an OT oracle.

[5]Combining our work with results from [Hai08, GMW91], we can also obtain computational security against malicious adversaries in both the nonadaptive and adaptive settings.

Condition (3) applies when $t \geq n/2$ and when A and B do not have an OT edge between them. This condition is effectively the converse of the impossibility result of [HIK07], which states that any n-party OT graph whose complement contains $K_{n-t,n-t}$ as a subgraph cannot allow a n-party secure computation that tolerates t semihonest corruptions. Condition (3) implies that any n-party OT graph whose complement does not contain $K_{n-t,n-t}$ as a subgraph can run n-party secure computations tolerating t semihonest corruptions.

Applying our main theorem. We first compare our positive results to those of [HIK07]. They investigate how to construct an OT graph with the minimum number of edges allowing n parties to execute a secure computation protocol. They show a construction for a graph with $(n + o(n))\binom{\lceil 1/\delta \rceil}{2}$ edges which they prove is sufficient for resilience against an adversary that corrupts $(1 - \delta)n$ parties. Our result provides a complete, simple characterization of which OT graphs on n vertices are sufficient to run a t-secure protocol generating OT correlations between all pairs of vertices for any $t \geq n/2$, which is sufficient to obtain a protocol for secure computation among the n parties [Kil88, IPS08]. Our main theorem also implies that determining the minimum number of OT edges needed to execute a secure computation protocol for general $n, t \geq n/2$ is equivalent to an open problem in graph theory posed by Zarankiewicz in 1951 [KST54].

Our results immediately imply that for some values of t, extremely simple sparse OT graphs suffice for achieving secure multiparty computation. For n even and $t = n/2$, we have that the t-claw graph (cf. Fig. 3-4a) has t edges and suffices to achieve t-secure multiparty computation. For n odd and $t = (n+1)/2$, the $(t+1)$-cycle has $t+1$ edges and suffices to achieve t-secure multiparty computation. We show in Appendix 3.9 that these examples are the sparsest possible graphs which can achieve $\lfloor (n+1)/2 \rfloor$-secure multiparty computation.

Next, our results are also well-suited to make use of an OT infrastructure for secure computation. Specifically, let G_I denote the OT graph consisting of existing OT edges between parties that are part of the infrastructure. Now suppose a pair of parties A, B not connected by an OT edge wish to execute a secure computation protocol. Then they can find a subgraph G of G_I with $A, B \in G$ and $|G| = n$ such that they agree that at most t out of the n parties can be corrupt and the partition condition in our main theorem holds for G. Since it is possible to handle a dishonest majority, parties do not have to settle for a lower threshold and can enjoy increased confidence in the security of their protocol by making use of the infrastructure. Surprisingly, it turns out the OT subgraph G need not even contain t OT edges to offer resilience against t corruptions (cf. Fig. 3-2(c) with $n = 4, t = 2$).

A pair of parties may use the OT correlations generated as the base OTs for an OT extension protocol and inexpensively generate many OT correlations that can be saved for future use or to add to the OT infrastructure. In any case, it should be clear that our protocols readily allow load-balancing across the OT infrastructure and are also abort-tolerant in the sense that if some subgraph G ends up not delivering the output, then one can readily use a different subgraph G'. Thus we believe that our results can be used to build a *scalable* infrastructure for secure computation that allows (1) amortization, (2) routing, and (3) is robust.

An important caveat regarding efficiency. In the special cases $t = n/2 + O(1)$ and $t = n - O(1)$, determing whether a graph satisfies the partition condition requires at most $\mathsf{poly}(n)$ time. However, in general the problem is coNP-complete, since it can be restated in the graph complement as subgraph isomorphism of a complete bipartite graph [GJ79]. Our protocols are efficient in n only for $t = n/2 + O(1)$ and $t = n - O(1)$.[6] In particular, our protocol is quite efficient for small values of n, a setting in which computing OT correlations in the presence of a dishonest majority may be especially useful in practice.

Organization After discussing preliminaries in Section 5.2, we give an overview of some of our techniques in Section 3.3, where we show solutions for the simple case when $n = 4$ and $t = 2$. Following this we briefly sketch the lower bound in Section 3.4 and describe the building blocks required for our upper bounds in Section 3.5. The rest of the paper is devoted to proving the upper bound first for the specific cases of $t = n/2$ (Section 3.6) and $t = n - 2$ (Section 3.7). We then use each of these protocols in two different solutions for the

[6]For $t = n/2 + O(1)$, we achieve efficiency using computationally-secure OT extension (e.g. [Bea96, IKNP03]). Our protocol with information-theoretic security is quasipolynomial-time for $t = n/2 + O(1)$. We do, however, achieve information-theoretic security in polynomial time for $t = n - O(1)$.

general case of $t \geq n/2$ in Section 3.8 which are efficient for different values of the corruption threshold t.

3.2 Preliminaries

3.2.1 Notation and definitions

Let $\mathcal{X}, \mathcal{Y}$ be two probability distributions over some set S. Their *statistical distance* is

$$\mathbf{SD}(\mathcal{X}, \mathcal{Y}) \stackrel{\text{def}}{=} \max_{T \subseteq S} \{\Pr \mathcal{X} \in T - \Pr \mathcal{Y} \in T\}$$

We say that $\mathcal{X}$ and $\mathcal{Y}$ are ε-close if $\mathbf{SD}(\mathcal{X}, \mathcal{Y}) \leq \varepsilon$ and this is denoted by $\mathcal{X} \approx_\varepsilon \mathcal{Y}$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are identical if $\mathbf{SD}(\mathcal{X}, \mathcal{Y}) = 0$ and this is denoted by $\mathcal{X} \equiv \mathcal{Y}$.

All graphs addressed in this work are undirected. We denote a graph as $G = (V, E)$ where V is a set of vertices and E is a set of edges. We denote an edge e as $e = \{v_1, v_2\}$, where $v_1, v_2 \in V$.

For $n \in \mathbb{N}$, let K_n denote the complete graph on n vertices. Let Λ_a^s denote the graph $G = (V, E)$ on $2a + s$ vertices with $V = V_A \dot{\bigcup} V_S \dot{\bigcup} V_B$, where $|V_A| = |V_B| = a$ and $|V_S| = s$, and

$$E = \{\{v_1, v_2\} : v_1 \notin V_A \vee v_2 \notin V_B\}$$

We will sometimes consider subgraphs of Λ_a^s which preserve labels of vertices. In this case we will always label the vertices so that vertex $A \in V_A$ and vertex $B \in V_B$.

For two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with the same vertex set V, we say that G_1 and G_2 are *$(v_1, \ldots, v_\ell)$-isomorphic*, denoted by $G_1 \simeq_{v_1, \ldots, v_\ell} G_2$, if the two graphs are isomorphic to one another while fixing the labelings of vertices $v_1, \ldots, v_\ell \in V$, that is, there exists an isomorphism σ such that $\sigma(v_i) = v_i$ for all $i \in [\ell]$.

Similarly, given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \subseteq V_2$ and $v_1, \ldots, v_\ell \in V_1$, we say that G_1 is a *$(v_1, \ldots, v_\ell)$-subgraph* of G_2, denoted $G_1 \subseteq_{v_1, \ldots, v_\ell} G_2$, if G_1 is $(v_1, \ldots, v_\ell)$-isomorphic to some subgraph of G_2.

In particular, in the special case that graph $G = (V, E)$ contains vertices $A, B \in V$, we say that G is an *(A, B)-subgraph* of Λ_a^s (or that $G \subseteq_{A,B} \Lambda_a^s$) if there is an isomorphism σ between G and a subgraph of Λ_a^s such that A is mapped into set V_A and B is mapped into set V_B (that is, $\sigma(A) \in V_A$ and $\sigma(B) \in V_B$).

Call an n-vertex graph $G = (V, E)$ *k-unsplittable* for $k \leq n/2$ if any two disjoint sets of k vertices have some edge between them. That is, G is k-unsplittable if for all partitions of the vertices V into three disjoint sets V_1, V_2, V_3 of sizes $|V_1| = |V_2| = k$ and $|V_3| = n - 2k$, there exists some edge $(u, v) \in E$ with $u \in V_1, v \in V_2$. It is immediate from this definition that G is k-unsplittable if and only if $G \not\subseteq \Lambda_k^{n-2k}$.

Similarly, call G *(k, A, B)-unsplittable* for $k \leq n/2$ and $A, B \in V$ if any two disjoint sets of k vertices containing A and B, respectively, have some edge between them. That is, G is (k, A, B)-unsplittable if for all partitions of the vertices of V into three disjoint sets V_1, V_2, V_3 of sizes $|V_1| = |V_2| = k$ and $|V_3| = n - 2k$ such that $A \in V_1$ and $B \in V_2$, there exists some edge $(u, v) \in E$ with $u \in V_1, v \in V_2$. From this definition we have immediately that G is (k, A, B)-unsplittable if and only if $G \not\subseteq_{A,B} \Lambda_k^{n-2k}$.

3.2.2 Secure Computation

Consider the scenario of n parties $P_1, \ldots, P_n$ with private inputs $x_1, \ldots, x_n \in \mathcal{D}$ computing a function $f : \mathcal{D}^n \to \mathcal{D}^n$. Let Π be a protocol computing f. We consider security against adaptive t-threshold adversaries, that is, adversaries that adaptively corrupt a set of at most t parties, where $0 \leq t < n$.[7] We assume the adversary to be semihonest (i.e. honest-but-curious). That is, the corrupted parties follow the prescribed protocol, but the adversary may try to infer additional information about the inputs of the honest parties. As noted in [HIK07], in the computational setting, using zero-knowledge proofs, it is possible to generically compile a protocol which is secure against semihonest adversaries into another protocol which is secure against adaptive malicious adversaries [GMW91].[8] This justifies our focus

[7]Note that when $t = n$, there is nothing to prove.

[8]We note that in the computational setting, it is also possible to transform, in a *black-box* way, a protocol which is secure against semihonest adversaries into another protocol which is secure against static malicious

on the semihonest setting here.

For a PPT adversary $\mathcal{A}$, let random variable $\mathrm{REAL}_{\Pi,\mathcal{A}}^{x_1,\dots,x_n}$ consist of the views of the corrupted parties when the protocol Π is run on parties $P_1, \dots, P_n$ with inputs $x_1, \dots, x_n$ respectively. In the ideal world, the honest parties are replaced with a simulator $\mathcal{S}$ that does not receive input values and knows only the output value of each corrupted party in an honest execution of the protocol. We define the random variable $\mathrm{IDEAL}_{\Pi,\mathcal{A},\mathcal{S}}^{x_1,\dots,x_n}$ as the output of the adversary $\mathcal{A}$ in the ideal game with the simulator when the inputs to parties $P_1, \dots, P_n$ are $x_1, \dots, x_n$, respectively.

Definition 3.2.1. *A protocol Π is said to t-securely compute the function f if*

- *For all $x_1, \dots, x_n \in \mathcal{D}^n$, party P_i receives y_i, where $(y_1, \dots, y_n) = f(x_1, \dots, x_n)$, at the end of the protocol.*
- *For all adaptive semihonest PPT t-threshold adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for all $x_1, \dots, x_n \in \mathcal{D}^n$*

$$\{\mathrm{REAL}_{\Pi,\mathcal{A}}^{x_1,\dots,x_n}\} \equiv \{\mathrm{IDEAL}_{\Pi,\mathcal{A},\mathcal{S}}^{x_1,\dots,x_n}\}$$

This definition is for secure computation with perfect information-theoretic security and a nonadaptive adversary. By [CDD+01], in the semihonest setting with information-theoretic security, any protocol which is nonadaptively secure is also adaptively secure. Consequently, satisfying this definition suffices to achieve adaptive security.

In the discussion below, we will sometimes relax security to statistical or computational definitions. A protocol is statistically t-secure if the random variables $\mathrm{REAL}_{\Pi,\mathcal{A}}^{x_1,\dots,x_n}$ and $\mathrm{IDEAL}_{\Pi,\mathcal{A},\mathcal{S}}^{x_1,\dots,x_n}$ are statistically close, and computationally t-secure if they are computationally indistinguishable.

3.2.3 Oblivious Transfer

In this work OT refers to 1-out-of-2 oblivious transfer defined as follows.

adversaries [Hai08].

Definition 3.2.2. *We define 1-out-of-2 oblivious transfer f_{OT} for a sender $A = P_1$ with inputs $x_0, x_1 \in \{0,1\}^m$, a receiver $B = P_2$ with input $b \in \{0,1\}$ and $n-2$ parties $P_3, \dots, P_n$ with input $\perp$ as*

$$f_{\mathrm{OT}}((x_0, x_1), b, \perp, \dots, \perp) = (\perp, x_b, \perp, \dots, \perp)$$

Note that while OT is typically defined as a 2-party functionality, the definition above adapts it our setting and formulates OT as an n-party functionality where only two parties supply non-$\perp$ inputs.

Definition 3.2.3. *Let G be a network consisting of n parties $A = P_1, B = P_2, P_3, \dots, P_n$. Then a t-secure OT protocol $\Pi^{G,t}_{A \to B}$ is a protocol that t-securely computes the function f_{OT} on the inputs of the parties with A as the sender and B as the receiever.*

We note that OT is symmetric, in the following sense.

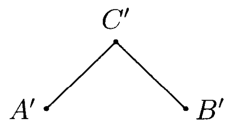
Lemma 3.2.1. *[WW06] If there exists a t-secure OT protocol $\Pi^{G,t}_{A \to B}$ for an n-party network G with n parties $A = P_1, B = P_2, P_3, \dots, P_n$ with A as the sender and B as the receiever, then there exists a t-secure OT protocol $\widehat{\Pi}^{G,t}_{B \to A}$ for the same n parties with B as the sender and A as the receiever.*

We represent parties as nodes of a graph G where an edge $\{A, B\}$ indicates that parties A and B may run a 1-secure OT protocol with A as the sender and B as the receiver. By Lemma 3.2.1, the roles of the sender and receiver may be reversed, so it makes sense to define G as an undirected graph.

We note the following result regarding the completeness of OT for achieving arbitrary secure multiparty computation.

Lemma 3.2.2. *[Kil88, GV87, IPS08] Consider the complete network $G \simeq K_n$ on n vertices. Then, for any function $f : D^n \to R^n$, there exists a protocol Π which $(n-1)$-securely computes f, where party i receives the ith input $x_i \in D$ and produces the ith output $(f(x))_i \in R$.*

(a) G_{CK}



(b) G_{HIK}

Figure 3-1: Known impossibility results. Securely computing f_{OT} between A' and B' is impossible for $t = 1$ in G_{CK} and is impossible for $t = 2$ in G_{HIK}.

3.3 Warm-ups

Let $G = (V, E)$ be an n-vertex graph representing a network with n parties, where an edge $\{P_i, P_j\} \in E$ indicates that parties P_i and P_j may run a 1-secure 2-party OT protocol with P_i as the sender and P_j as the receiver. Let $t < n$ be an upper bound on the number of corruptions made by the adversary. The central question considered in this work is the following. For which graphs G and which pairs of parties $A, B \in V$ does there exist a t-secure OT protocol with A as the sender and B as the receiver?

We begin by discussing some simple special cases of small networks. These will provide useful intuition for our main results. For $t < n/2$, it is possible to obtain a t-secure OT protocol for any n-vertex graph $G = (V, E)$ between any $A, B \in V$, since we can perform secure multiparty computation without any pre-existing OT channels if there is an honest majority [RB89]. It remains to consider the setting where $t \geq n/2$.

A few small cases have been resolved in prior work. For $n = 2$, $t = 1$, a 1-secure OT protocol (with perfect security) between the vertices of the two-vertex graph G does not exist unless the parties were already connected by an OT channel [CK89, Kus89]. This result is illustrated in Figure 3-1a.

For $n = 3$, $t = 2$, it is known that we can obtain a 2-secure OT protocol between a pair of vertices A, B only if those vertices are already connected by an OT channel, even if there are OT channels from both A and B to the third vertex C as depicted in Figure 3-1b. More generally, for any $n \geq 2$ and $t = n - 1$, there exists a t-secure OT protocol with sender A and receiver B only if those vertices are already connected by an OT channel, even if all other $\binom{n}{2} - 1$ pairs of vertices are connected by OT channels [HIK07]. This also resolves the

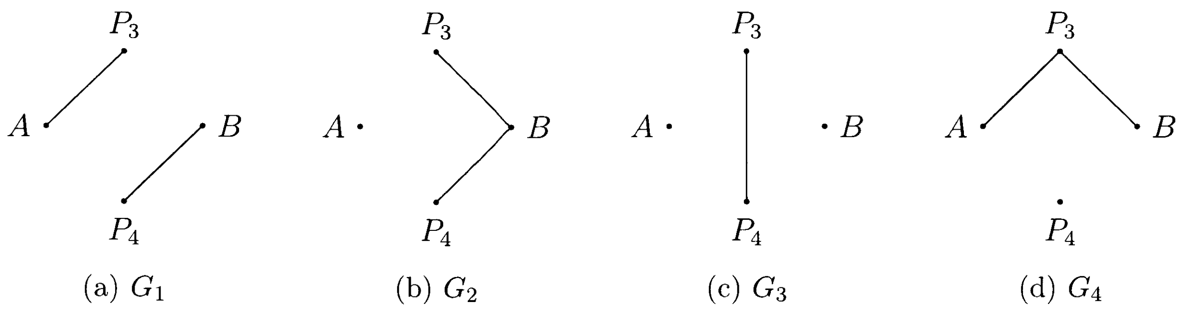


Figure 3-2: Cases for $n = 4$ parties with $t = 2$ corruptions.

question for $n = 4, t = 3$.

The remainder of this section is devoted to an exploration of the setting $n = 4, t = 2$. This is the smallest case not resolved by prior techniques, and will illustrate many of the tools used in subsequent sections to obtain our general protocols. The key cases for $n = 4, t = 2$ are shown in Figure 3-2. As discussed below, these cases are sufficient to completely resolve the four-party setting.

3.3.1 Case 1 : Figure 3-2a

We first show that if $G \simeq_{A,B} G_1$ then there does not exist a 2-secure OT protocol for G with A as the sender and B as the receiver.[9] This is a consequence of the impossibility result of [CK89, Kus89]. An outline of the argument is as follows.

Consider components $C_1 = \{A, P_3\}$ and $C_2 = \{B, P_4\}$ of G, and let Π be a 2-secure protocol computing f_{OT} in G with A as the sender and B as the receiver. Then we can use Π to construct a 1-secure protocol Π' for the 2-party network G_{CK} in Figure 3-1a with A' as the sender and B' as the receiver. In protocol Π', party A' runs Π for both parties of component C_1 of G, and B' runs Π for both parties of component C_2. OT channel invocations can be handled locally, since all OT channels in G are between parties in the same component. Since protocol Π is 2-secure, in particular it is secure against corruptions of parties in C_1 or the parties in C_2. Consequently Π' is a 1-secure OT protocol for a network

[9]Recall that $H \simeq_{A,B} H'$ for two graphs H, H' if there exists an isomorphism between H and H' preserving the labels of vertices A and B.

$G' \simeq_{A',B'} G_{\mathrm{CK}}$ with A' as the sender and B' as the receiver. However, from [CK89, Kus89], we know that no such protocol exists with perfect security. Consequently there is no 2-secure protocol Π for a network $G \simeq_{A,B} G_1$.

Note that this impossibility holds not only for $G \simeq_{A,B} G_1$ but for any (A, B)-subgraph of G_1. In particular, if $G = (V, E)$ is a four-vertex graph a single edge that is incident to vertex A or vertex B, then G cannot have a 2-secure protocol computing f_{OT} between A and B except in the trivial case when there is already an edge $\{A, B\} \in E$. This technique of reducing to the known impossiblity results of [CK89, Kus89, HIK07] to obtain lower bounds is described formally in Section 3.4.

3.3.2 Case 2 : Figure 3-2b

In this example we obtain a positive result, showing that there exists a 2-secure OT protocol with A as the sender and B as the receiver. Since B has degree 2 in G_2, we have that either B or one of its neighbors must be honest, and so one of the two OT channels must contain an honest party. This suggests the idea of using secret-sharing to ensure security against 2 corruptions.

Consider the following OT protocol where sender A has inputs $x_0, x_1 \in \{0,1\}^m$ and receiver B has input $b \in \{0,1\}$. A computes 2-out-of-2 shares (x_0^1, x_0^2) and (x_1^1, x_1^2) of its inputs x_0, x_1, respectively. A then sends shares x_0^1 and x_1^1 to party P_3 and x_0^2 and x_1^2 to party P_4. Parties P_3 and B invoke their secure OT channel with inputs (x_0^1, x_1^1) and b, and parties P_4 and B invoke their secure OT channel with inputs (x_0^2, x_1^2) and b respectively. B uses the obtained shares x_b^1, x_b^2 to reconstruct x_b.

We informally argue the 2-security of this protocol assuming that exactly one of A and B is corrupt.[10] Consider the case where A is corrupt and B is honest. The input of B is

[10]An additional step is needed to address the case in which P_3 and P_4 are corrupt and A and B are both honest. Then P_3 and P_4 can learn x_0 and x_1, the inputs of A, in the protocol just described. This can be handled with the technique of OT correction, using a one-time pad and the secure point-to-point channel between A and B. Equivalently, we could run the protocol on random inputs, and then use method of [Bea95] to obtain 1-out-of-2 OT from random OT. If A and B are both corrupt then there is nothing to prove.

only used over secure OT channels, so by the 1-security of the OT channels with P_3 and P_4, the corrupt parties can learn nothing about B's input bit b. Now consider the case where B is corrupt and A is honest. Either P_3 or P_4 must be honest. If P_3 is honest then the security of OT channel $\{P_3, B\}$ implies that B learns nothing about share x^1_{1-b}, so the security of the secret sharing scheme implies that the corrupt parties do not use x_{1-b}. By symmetry, the same argument applies if P_4 is honest. This completes the argument.

Note that by Lemma 3.2.1, we can also obtain a 2-secure OT protocol from A to B whenever A has degree 2 in OT network. Furthermore, we can extend this idea to construct a t-secure OT protocol whenever either the sender or the receiver has degree at least t. We call this protocol the t-claw protocol and describe it in detail in Section 3.5.1.

3.3.3 Case 3 : Figure 3-2c

Somewhat surprisingly, we can also show a positive result for graphs $G \simeq_{A,B} G_3$ even though the OT network has no edges involving either the sender A or the receiver B. The protocol is as follows. Since parties P_3 and P_4 have an OT channel between them, by Lemma 3.2.2, they can perform 1-secure MPC between them. P_3 and P_4 use MPC to compute 2-out-of-2 shares of OT correlations with uniformly random inputs and send corresponding shares to A and B, who can then reconstruct the correlations. More concretely, the MPC protocol computes 2-out-of-2 shares (r_0^1, r_0^2), (r_1^1, r_1^2) of two randomly sampled m-bit strings r_0, r_1, 2-out-of-2 shares (c^1, c^2) of a random bit $c \in \{0,1\}$, and independent 2-out-of-2 shares (s^1, s^2) of the string r_c. Party P_3 receives the first share of each secret, and party P_4 receives the second share. Party P_3 then sends shares r_0^1, r_1^1 to A and s^1, c^1 to B, while P_4 sends shares r_0^2, r_1^2 to A and s^2, c^2 to B. A can then reconstruct r_0 and r_1, and B can reconstruct c and r_c. Parties A and B have now established a random OT correlation, which they can use to perform OT with their original inputs using OT correction [Bea95].[11]

We now informally argue the 2-security of this protocol. If A and B are both honest, then the corrupt parties receive no information about their inputs, while if A and B are both

[11]This OT correction step can be performed as follows. Party B sends $b' = b \oplus c$ to A. A responds with $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$. Finally, B computes $y_b \oplus r_c = x_b$.

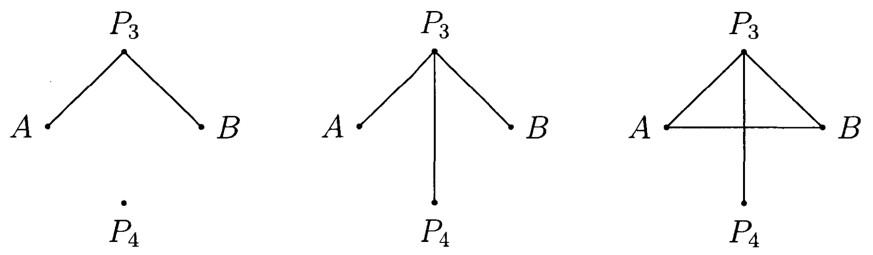


Figure 3-3: Illustrating the cascading protocol for Case 4 : Figure 3-2d; (a) $\rightarrow$ (b) $\rightarrow$ (c)

corrupt then there is nothing to prove. Consequently we can assume that exactly one of A and B is corrupt and that either P_3 or P_4 is honest. If A is corrupt and P_3 or P_4 is honest, then the adversary learns nothing about c and r_c, since it only sees one of the two shares of each. The OT correction phase uses these strings as one-time pads for inputs which are unknown to the adversary, and consequently are information-theoretically hidden from the adversary. Consequently A learns nothing about B. The case where B is corrupt and P_3 or P_4 is honest follows by the same argument.

This construction can be extended to obtain a t-secure OT protocol whenever the OT graph contains a t-clique consisting of t parties which are not the OT sender or receiver. We call this protocol the t-clique protocol and describe it in detail in Section 3.5.2.

3.3.4 Case 4 : Figure 3-2d

We also obtain a positive result for graphs $G \simeq_{A,B} G_4$. We introduce here a technique we call cascading. The idea is as follows. Using the protocol described in Section 3.3.2 for network G_2 of Figure 3-2b, we have 2-secure OT protocol with P_3 as the sender and P_4 as the receiver. This effectively gives us an OT channel between P_3 and P_4. Applying the protocol from Section 3.3.3 on the augmented network, we obtain a 2-secure OT protocol with A as the sender and B as the receiver. We describe this pictorially in Figure 3-3.

The 2-security of the protocol follows from the 2-security of the underlying protocols of Sections 3.3.2 and 3.3.3. The technique of cascading for combining t-secure protocols is described in detail in Section 3.5.3.

3.3.5 Cases 1–4 are exhaustive

Note that a t-secure OT protocol with sender A and receiver B in an OT network G trivially yields a t-secure protocol for any network G' such that $G \subseteq_{A,B} G'$. From cases 1 and 3, we can securely compute f_{OT} in a network G containing at most a single edge if and only if the edge is $\{A, B\}$ or $\{P_3, P_4\}$. From cases 1, 2, and 4, we can compute f_{OT} in a network G containing two or more edges including neither of $\{A, B\}$ or $\{P_3, P_4\}$ if and only if there is some vertex with degree at least 2 in the OT graph. This completes the characterization of 4-party networks with 2 corruptions.

3.4 Lower Bound

We now describe a family of impossibility results using a generic reduction to the impossiblity result in [HIK07], which we restate in our language below.

Lemma 3.4.1. *[HIK07] Consider any three party network G with $G \simeq_{A',B'} G_{\mathrm{HIK}}$, the graph in Figure 3-1b. Then any 2-secure OT protocol with A' as the sender and B' as the receiver can be used (as a black box) to obtain a 1-secure OT protocol for a network G' with $G' \simeq_{A',B'} G_{\mathrm{Kus}}$, the graph in Figure 3-1a, with A' as the sender and B' as the receiver.*

The theorem below describes an impossibility result over a family of networks. We note that this result was observed in [HIK07]; we restate it our language and provide a formal proof.

Theorem 3.4.2. *Let $n \geq 2$ and $n/2 \leq t < n$, and let G be an n party network such that $G \subseteq \Lambda_{n-t}^{2t-n}$, with $P_1 \in V_A$ and $P_2 \in V_B$. Any t-secure OT protocol for G with P_1 as the sender and P_2 as the receiver can be used (as a black box) to obtain a 1-secure OT protocol for a network G' with $G' \simeq_{A,B} G_{\mathrm{CK}}$ with A' as the sender and B' as the receiver.*

Proof. Let G be an n party network with $G = (V, E)$ such that $G \subseteq \Lambda_{n-t}^{2t-n}$. Then, we may write $V = V_A \dot{\bigcup} V_S \dot{\bigcup} V_B$, where $|V_A| = |V_B| = n - t$ and $|V_S| = 2t - n$, with $P_1 \in V_A$ and

$P_2 \in V_B$ and $E(V_A, V_B) = \varnothing$, where $E(V_A, V_B)$ represents the set of edges with one endpoint in V_A and the other in V_B.

Let Π be a t-secure OT protocol for G with P_1 as the sender and P_2 as the receiver. If $t > n/2$, then we can use Π to construct a 2-secure OT protocol Π' for any three party network G' with $G' \simeq_{A',B'} G_{\text{HIK}}$ with A' as the sender and B' as the receiver below. Combining this with Lemma 3.4.1, the conclusion follows. We describe the construction of Π' below. If $t = n/2$, then we can use Π to construct a 1-secure OT protocol Π'' for any two party network G'' with $G'' \simeq_{A'',B''} G_{\text{CK}}$ with A'' as the sender and B'' as the receiver. The construction of Π'' is exactly the same as that of Π' and hence we omit its description here.

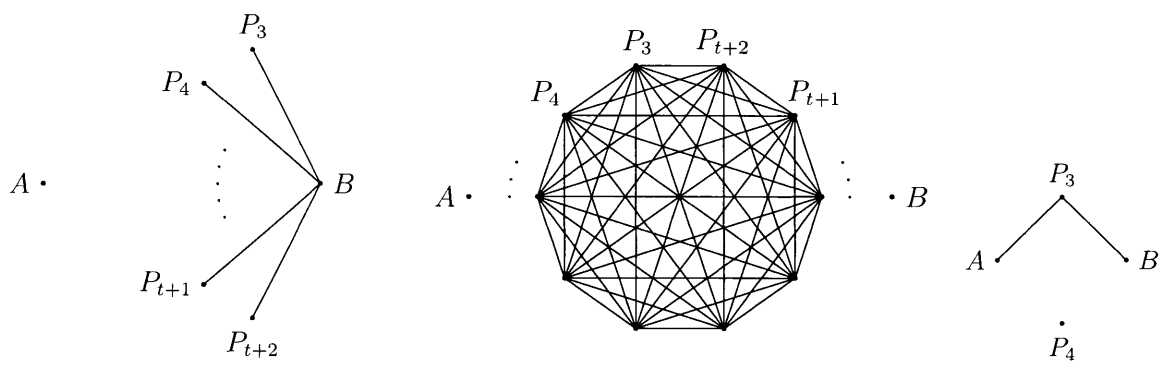
In protocol Π', party A' simulates the parties of component V_A, party C' simulates the parties of component V_S, and party B' simulates the parties of component V_B. Executions of 1-secure OT protocols between parties of the same component are handled locally and executions of 1-secure OT protocols between parties in different components are handled as follows:

- If the parties are in components V_A and V_S, then executions of 1-secure OT protocols between the parties are carried out using the OT edge $\{A', C'\}$ in the network G'.
- If the parties are in components V_B and V_S, then executions of 1-secure OT protocols between the parties are carried out using the OT edge $\{B', C'\}$ in the network G'.

Since $G \subseteq \Lambda_{n-t}^{2t-n}$, there are no executions of 1-secure OT protocols between parties in components V_A and V_B in protocol Π.

Correctness of Π' is obvious. We now prove 2-security of Π'. Intuitively, since Π is t-secure, in particular, it is secure against corruptions of parties $V_A \cup V_S$ or the parties $V_B \cup V_S$. Consequently protocol Π' is secure against corruptions $\{A', C'\}$ or $\{B', C'\}$ and hence Π' is a 2-secure protocol.

Formally, we prove this as follows. Since Π is t-secure, there exists a simulator S such that for every PPT t-threshold adversary $\mathcal{A}$, $\left\{\text{REAL}_{\Pi,\mathcal{A}}^{(x_0,x_1),b,\perp,\ldots,\perp}\right\} \equiv \left\{\text{IDEAL}_{\Pi,\mathcal{A},\mathcal{S}}^{(x_0,x_1),b,\perp,\ldots,\perp}\right\}$, where $x_0, x_1 \in \{0,1\}^m$ and $b \in \{0,1\}$. The simulator S' for the protocol Π' behaves exactly the same as S while simulating the parties of component V_A as A', those of of component V_B as



(a) G^t_{claw} (b) G^t_{clique} (c) $G^2_{\text{2-path}}$

Figure 3-4: Building block networks. (a) t-claw graph (b) t-clique graph (c) 2-path graph

B' and those of of component V_S as C'. It is easy to see that for any 2-threshold adversary $\mathcal{A}'$, namely, one which corrupts $\{A', C'\}$ or the one which corrupts $\{B', C'\}$, $\left\{\text{REAL}^{(x_0,x_1),b,\perp}_{\Pi',\mathcal{A}'}\right\} \equiv \left\{\text{IDEAL}^{(x_0,x_1),b,\perp}_{\Pi',\mathcal{A}',\mathcal{S}'}\right\}$ since Π is secure against corruptions of parties $V_A \cup V_S$ or the parties $V_B \cup V_S$. $\square$

3.5 Building Blocks

In this section, we describe a few key protocols and techniques that we use in the subsequent sections to prove our main theorem.

3.5.1 The t-claw Protocol

The first protocol we describe is the t-claw protocol, where the graph G describing the network is such that $G \simeq_{A,B} G^t_{\text{claw}}$. The protocol is described in Protocol 1. The protocol is a straightforward generalization of the one described in Section 3.3.2. The idea is for A to compute t-out-of-t shares of its inputs and distribute them among the t parties connected to B. These t parties then perform OT with B so that B receives the shares to reconstruct his output.

Lemma 3.5.1. *Protocol 1 is an efficient t-secure OT protocol for a network $G \simeq_{A,B} G^t_{\text{claw}}$ with A as the sender and B as the receiver.*

Protocol 1: t-claw Protocol

Preliminaries: Let $A, B, P_3, \dots, P_{t+2}$ be the $t+2$ parties in a network $G \simeq_{A,B} G^t_{\text{claw}}$. A has inputs $x_0, x_1 \in \{0,1\}^m$ and B has input $b \in \{0,1\}$.

Protocol:

1. B chooses a random bit $c \in \{0,1\}$ and sends $b' = b \oplus c$ to A.
2. A chooses two random one-time pads $r_0, r_1 \in \{0,1\}^m$ and sends $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ to B.
3. A then computes t-out-of-t shares $(r_0^1, \dots, r_0^t)$ and $(r_1^1, \dots, r_1^t)$ of r_0, r_1, respectively.
4. For each $i \geq 3$, A sends shares r_0^i and r_1^i to party P_i.
5. For each $i \geq 3$, parties P_i and B invoke the OT protocol $\Pi^{G,1}_{P_i \to B}$ with inputs (r_0^i, r_1^i) and c respectively.
6. B uses the obtained shares $r_c^1, \dots, r_c^t$ to reconstruct r_c.
7. B finally computes $y_b \oplus r_c = x_b$.

Proof Intuition. The t-security of the procotol can be seen as follows. Steps 1, 2 and 7 perform OT correction, that is, they perform a transformation from random OT to 1-out-of-2 OT. This transformation protects against the case that the parties $P_3, \dots, P_{t+2}$ (that is, all but A and B) are corrupt. Suppose A were corrupt and B were honest. Clearly, A colluding with any of the parties $P_3, \dots, P_{t+2}$ provides A with no additional information since all they possess are shares sent by A. Next, if A were honest and B corrupt, at least one of the parties $P_3, \dots, P_{t+2}$ must be honest. B has no information about those shares and hence does not learn anything. Finally, if both A and B were corrupt, there is nothing to prove.

Proof. Let A be a t-threshold adversary which corrupts parties T, $|T| \leq t$. We will construct a simulator S which plays the role of the uncorrupted parties. If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so S can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \varnothing$ then S chooses arbitrary inputs x_0, x_1, b and runs the protocol, invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since corrupted parties only learn secret shares of independently random values, the view of the adversary is independent of the choice of x_0, x_1, b and is identical to the real world.

Otherwise, we have that the corrupted parties T include exactly one of A, B. If $A \in T$ but $B \notin T$, then S chooses arbitrary input b and runs the protocol, invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since the OT simulator does not reveal the input c, and since the adversary only learns the direct sum of b with the random bit c, the view of the adversary is identical regardless of the value of b and in particular is identical to the real world.

Finally we have the case $B \in T, A \notin T$. Here the simulator S is given the output value x_b. S runs the protocol with (x_b, x_b) as the input to A, again invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since $|T| \leq t$ and $B \in T$, at most $t-1$ of the t parties $P_3, \ldots, P_{t+2}$ are corrupted. Consequently the adversary observes at most $t-1$ shares of the random one-time pads r_0, r_1, so by the security of t-out-of-t secret sharing, conditioned on the remaining shares being hidden, the distribution of the observed shares is independent of r_0, r_1. The adversary learns the shares of r_c, but by the security of the OT channels, the view of the adversary in step 3 and onward is independent of the remaining shares of r_{1-c} and consequently is independent of the choice of r_{1-c}. Consequently the view of the adversary in step 3 and onward is independent of r_{1-c}. In step 2, the adversary sees $y_b = x_b \oplus r_c$ and $y_{1-b} = x_{1-b} \oplus r_{1-c}$, so by the security of the one-time pad, the view of the adversary is independent of x_{1-b}. Consequently the overall view of the adversary is identical in the real and ideal worlds. □

3.5.2 The t-clique Protocol

The next protocol we describe is the t-clique protocol, where the graph G describing the network is such that $G \simeq_{A,B} G^t_{\text{clique}}$. The protocol is described in Protocol 2. The protocol is a straightforward generalization of the one described in Section 3.3.3. The idea is for the parties $P_3, \ldots, P_{t+2}$ to compute t-out-of-t shares of OT correlations and send them to A and B respectively. The parties have a complete network of OT channels, so this can be done via multiparty computation (Lemma 3.2.2). A and B then perform OT correction using their secure channel.

Lemma 3.5.2. *Protocol 2 is an efficient t-secure OT protocol for a network $G \simeq_{A,B} G^t_{\text{clique}}$ with A as the sender and B as the receiver.*

Protocol 2: t-clique Protocol

Preliminaries: Let $A, B, P_3, \ldots, P_{t+2}$ be the $t+2$ parties in a network $G \simeq_{A,B} G^t_{\text{clique}}$. A has inputs $x_0, x_1 \in \{0,1\}^m$ and B has input $b \in \{0,1\}$.

Protocol:

1. Parties $P_3, \ldots, P_{t+2}$ use their pairwise OT channels to run t-secure MPC for the function f using the protocol from Lemma 3.2.2 for the function f described ahead. The function f is to securely compute t-out-of-t shares $(r_0^1, \ldots, r_0^t)$, $(r_1^1, \ldots, r_1^t)$ of two randomly sampled one-time pad keys r_0, r_1, $(c^1, \ldots, c^t)$ of a random bit $c \in \{0,1\}$, and independent shares $(s^1, \ldots, s^t)$ of key r_c, so that party $i+2$ receives only shares r_0^i, r_1^i, s^i, c^i for each i.
2. Each party P_{i+2} for $i \geq 1$ sends shares r_0^i, r_1^i to A and s^i, c^i to B.
3. A uses shares $(r_0^1, \ldots r_0^t)$ and $(r_1^1, \ldots, r_1^t)$ to reconstruct r_0 and r_1.
4. B uses shares $(c^1, \ldots, c^t)$ and $(s^1, \ldots, s^t)$ to reconstruct c and r_c and sends $b' = b \oplus c$ to A.
5. A computes $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ and sends both to B.
6. B computes $y_b \oplus r_c = x_b$.

Proof Intuition. The t-security of the procotol can be seen as follows. Steps 4, 5 and 6 perform OT correction, that is, they perform a transformation from random OT to 1-out-of-2 OT. This transformation protects against the case that all of parties $P_3, \ldots, P_{t+2}$ (that is, all but A and B) are corrupt. If one of A and B were corrupt, there exists at least one honest party among the parties $P_3, \ldots, P_{t+2}$. Hence, even by colluding, A or B would have no information about those shares and would not learn anything. Finally, if both A and B were corrupt, there is nothing to prove.

Proof. Let A be a t-threshold adversary which corrupts parties T, $|T| \leq t$. We will construct a simulator S which plays the role of the uncorrupted parties. As above, if $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so S can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \varnothing$ then S chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend at all on the inputs are on point-to-point channels between A and B, the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that the corrupted parties T include exactly one of A, B. If $A \in T$ but $B \notin T$, then S chooses arbitrary input b and runs the protocol, invoking the MPC simulator for the protocol in step 1 (the existence of this simulator follows from Lemma

3.2.2). Since at least one of the parties $P_3, \ldots, P_{t+2}$ is uncorrupted, the security of the MPC protocol implies that the view of the adversary is independent of the uncorrupted parties' shares s^i and c^i, and so by the security of the secret sharing scheme is independent of the value of the bit c. The only message received by the adversary which depends on b is the bit $b' = b \oplus c$, so it follows that the view of the adversary is independent of the bit b and therefore is identical in the real and ideal worlds.

Finally we have the case $B \in T, A \notin T$. Here the simulator S is given the output value x_b. S runs the protocol with (x_b, x_b) as the input to A, again invoking the MPC simulator for the protocol in step 1. Since at least one of the parties $P_3, \ldots, P_{t+2}$ is uncorrupted, the security of the MPC protocol implies that the view of the adversary is independent of the uncorrupted parties' shares r_0^i and r_1^i, so by the security of t-out-of-t secret sharing, conditioned on the remaining shares being hidden, the distribution of the observed shares is independent of r_0, r_1. The adversary learns the shares of r_c, but by the security of the OT channels, the view of the adversary through step 4 is independent of the remaining shares of r_{1-c} and consequently is independent of the choice of r_{1-c}. In step 5, the adversary sees $y_b = x_b \oplus r_c$ and $y_{1-b} = x_{1-b} \oplus r_{1-c}$, so by the security of the one-time pad, the view of the adversary is independent of x_{1-b}. Consequently the overall view of the adversary is identical in the real and ideal worlds. $\square$

3.5.3 Cascading

The following building block is a generalization of the technique described in Section 3.3.4. The technique describes a general method of combining protocols iteratively. In our context, this can be thought of a tool for transforming a network described by a graph G to one described by a graph G', where $G \subseteq_V G'$ and G and G' are both graphs on the same vertex set V. In other words, it describes protocols as adding new edges indicating the establishment of OT correlations between new pairs of parties in the network. With this abstraction, it is easy to view the technique of cascading as one which combines protocols iteratively to transform the underlying network by adding new edges. This is described formally below.

Definition 3.5.1. *Let $G = (V, E)$ and $G' = (V, E')$ be two graphs on the same set of vertices, V, with $G \subseteq_V G'$. We say that a protocol Π* t-transforms *a network G into the network G' if for each $\{P_i, P_j\} \in E' \setminus E$, Π is a t-secure OT protocol for a network G with P_i as the sender and P_j as the receiver.*[12]

Lemma 3.5.3. *If Π_1 is a protocol that runs in time T_1 and t-transforms network G_1 into G_2, and Π_2 is a protocol that runs in time T_2 and t-transforms network G_2 into G_3, then there exists a protocol Π that runs in time T_1T_2 and t-transforms G_1 into G_3.*

Proof. The protocol Π simply runs Π_2, running protocol Π_1 to obtain the necessary correlations whenever Π_2 invokes OT on an edge of $G_2 \setminus G_1$. Let S_1 and S_2 be the simulators associated with Π_1 and Π_2 respectively. The simulator for Π simply runs S_2, invoking S_1 for OT calls made on edges in $G_2 \setminus G_1$. □

Using OT extension [Bea96, IKNP03], we can also obtain a computationally secure version of cascading with improved efficiency.

Lemma 3.5.4. *Let λ be a computational security parameter. Assuming one-way functions or correlation-robust hash functions, if Π_1 is a protocol that runs in time T_1 and t-transforms network G_1 into G_2, and Π_2 is a protocol that runs in time T_2 and t-transforms network G_2 into G_3, then there exists a computationally secure protocol Π that runs in time $\lambda \cdot T_1 + T_2 \cdot \mathsf{poly}(\lambda)$ and t-transforms G_1 into G_3.*

Proof. First, run protocol Π_1 λ times on random inputs to obtain λ independent OT correlations for each edge of $G_2 \setminus G_1$. Then run Protocol Π_2, using OT extension [Bea96, IKNP03] to obtain OT correlations for OT calls made on edges in $G_2 \setminus G_1$. □

3.5.4 The 2-path graph

The protocol described in this section is a commonly used subroutine in several of the protocols which follow. It is a particular combination of the tools encountered in Sections

[12]Note that a single protocol Π may set up independent random OT correlations for several pairs of parties $\{P_i, P_j\} \in E' \setminus E$. These correlations can be used to run 1-out-of-2 OT using OT correction.

Protocol 3: 2-path

Preliminaries: Let A, B, C, D be the parties, and let there exist OT channels (A, C) and (B, C). A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. Invoke Protocol 1 (2-claw) on parties (D, C, A, B) to obtain OT correlations on edge (D, C).
2. By Lemma 3.5.3, we have an OT channel between D and C.
3. Invoke Protocol 2 (2-clique) on parties (A, B, C, D).

3.5.1, 3.5.2 and 3.5.3. The subroutine, which we call 2-path, is the same as the one described in Section 3.3.4. It is used to obtain OT correlations between parties who have a common neighbor in a four-party network with at most two corruptions (see Figure 3-4c).

Lemma 3.5.5. *Protocol 3 is an efficient* 2*-secure OT protocol for a network* $G \simeq_{A,B} G^2_{\text{2-path}}$ *with* A *as the sender and* B *as the receiver.*

Proof. This follows immediately from Lemma 3.5.3 and the 2-security of Protocols 1 and 2 for $t = 2$ (Lemmata 3.5.1 and 3.5.2). □

3.5.5 Combiners

OT combiners aim to combine several insecure candidate protocols for establishing OT correlations between two parties into a single secure protocol. For a class of adversaries $\mathbb{A}$, it is possible to achieve this when the candidate protocols satisfy the property that a majority of them are secure against each advesary $A \in \mathbb{A}$. The following lemma is due to [MPW07, HIKN08], relying on prior work by [HKN$^+$05, Wul07] and based on a construction by [DKS99].

Lemma 3.5.6. *[MPW07, HIKN08] Let* $\mathbb{A}$ *be an adversary class. Suppose there exist* m *protocols* $\Pi_1, \ldots, \Pi_m$ *for* $f_{OT}(A, B, P_1, \ldots, P_n)$ *such that for any adversary* $A \in \mathbb{A}$ *a majority of the protocols are secure. Then, there exists a protocol* $\Pi^*(\Pi_1, \ldots, \Pi_m)$ *for* $f_{OT}(A, B, P_1, \ldots, P_n)$ *which is secure against all adversaries* $A \in \mathbb{A}$*. Moreover, if each protocol* Π_i *is efficient and perfectly secure, then so is* Π^*.

3.6 The case $t = n/2$

We now consider the specific case of $t = n/2$, that is, when at most half the parties are corrupt. We note that this is the smallest value of t for which the question is non-trivial. From the lower bounds proven in Theorem 3.4.2, we already have that for all n-party networks G containing A and B such that $G \subseteq_{A,B} \Lambda^0_{n/2}$, there exists no $n/2$-secure OT protocol with A as the sender and B as the receiver. Surprisingly Theorem 3.6.1 shows that these are the only networks for which $(n/2)$-secure OT between A and B is impossible. Below, we provide an explicit $n/2$-secure OT protocol between A and B whenever the network G is $(n/2, A, B)$-unsplittable.

Theorem 3.6.1. *Let G be an n-party network OT containing parties A and B. Then Protocol 5 is an $n/2$-secure OT protocol between A and B if and only if G is $(n/2, A, B)$-unsplittable.*

We analyze the efficiency of the protocol in Theorem 3.6.3 below. The protocol as stated runs in quasi-polynomial time. We can also obtain a computationally secure protocol which runs in polynomial time. The protocol we describe proceeds in two stages. In the first stage, the protocol transforms every connected component of the network into a clique. This transformation is very specific to the case of $t = n/2$, and in particular, for $t > n/2$ a connected component cannot in general function as a clique. This transformation is carried out by means of repeatedly calling Protocol 4, which obtains OT correlations between a pair of parties who have a common neighbour. This protocol uses the building block Protocol 3 from Section 3.5.4 along with machinery of OT combiners described in Section 3.5.5.

Lemma 3.6.2. *Let G be an n-vertex OT network with edges $\{A, C\}$ and $\{B, C\}$. Protocol 4 is an $n/2$-secure OT protocol for the network G with A as the sender and B as the receiver.*

Proof. We consider cases depending on the number of corrupted parties in the set $T = \{A, B, C\}$. If T contains at most one corrupted party, then each tuple (A, B, C, P_i) for $i \geq 4$ contains at most 2 corrupted parties, so each protocol Π_i in step 1 is secure. If T contains two corrupted parties, then there are at most $t - 2 = (n - 4)/2$ corrupted parties

Protocol 4: Completing Triangles

Preliminaries: Let $A, B, C, P_4, \ldots, P_n$ be the n parties, and let there exist OT channels (A, C) and (B, C). A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. Run a combined protocol $\Pi^*(\Pi_4, \ldots, \Pi_n)$ on the $n-3$ protocols $\Pi_4, \ldots, \Pi_n$, where
   - For each $i \geq 4$, Π_i denotes an invocation of Protocol 3 (2-path) with the four parties A, B, C, P_i with A as the sender and B as the receiver.

among $P_4, \ldots, P_n$, so a majority of these parties are honest. Consequently a majority of the protocols Π_i which are combined in step 1 are secure. Thus, in either case, by Lemma 3.5.6 the protocol is secure. Finally, if all three parties of T are corrupted, then all uncorrupted parties receive no input, so the simulator S can perfectly simulate the uncorrupted parties by running the honest protocol. Therefore Protocol 4 is $n/2$-secure. □

We now complete the proof of Theorem 3.6.1.

Proof Intuition (Theorem 3.6.1): It is easy to see that by invoking Protocol 4 repeatedly, one can obtain OT correlations between any pair of parties in the same connected component. In other words, using cascading (Lemma 3.5.3), we can assume that we are given a network which consists of disjoint cliques. This is done in step 1 of Protocol 5. Hence, if A and B were in the same connected component in G, this process would end up with correlations between A and B and we can terminate the protocol (step 2).

If A and B are in different components, then a natural next step is to run the clique protocol described in Section 3.5.2 with each of the cliques and parties A and B with the intent of setting up OT correlations between A and B. However, the number of corruptions t may be greater than the size of any clique, and so Protocol 2 may not be secure. However, for an invocation to be secure, we only require that the clique contains at least one honest party. A majority of parties must be in cliques containing at least one honest party, so if we invoke Protocol 2 for each of the parties on their respective cliques, for any adversary a majority of the invocations is secure. By Lemma 3.5.6 we can combine these candidate protocols to obtain a single secure protocol. This is performed in step 5 of Protocol 5. Finally, we note

Protocol 5: $n/2$ **corruptions**

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \dots, P_n$ be the n parties in a network $G = (V, E)$. A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. While there exist parties $P_i, P_j, P_k \in V$ such that $\{P_i, P_j\} \in E$, $\{P_j, P_k\} \in E$, but $\{P_i, P_k\} \notin E$:
   (a) Let S be the set of triples of distinct vertices $(X, Y, Z) \in V^3$ with $\{X, Y\} \in E, \{Y, Z\} \in E$, and $\{X, Z\} \notin E$.
   (b) For each triple $(X, Y, Z) \in S$, invoke Protocol 4 with independent random inputs $(r_0^{i,k}, r_1^{i,k})$ and $b^{i,k}$, to obtain OT correlations along edge $\{X, Z\}$.
   (c) Invoking cascading (Lemma 3.5.3), we can add $\{X, Z\}$ to the edge set E for all triples $(X, Y, Z) \in S$.

   The OT network G now consists of disjoint cliques $\mathcal{C}_1, \dots, \mathcal{C}_\ell$.
2. If A and B are in the same clique, then halt.
3. B samples a random bit c and sends $b' = b \oplus c$ to A.
4. A chooses random one-time pads r_0, r_1 and sends $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ to B.
5. Let $\mathcal{C}_1$ be the clique containing A and $\mathcal{C}_2$ be the clique containing B. For each party P_i, $i \geq 3$, let $\mathcal{C}(i)$ denote the clique containing party i, and let $P_{j_1}, \dots, P_{j_{|\mathcal{C}(i)|}}$ denote the parties in clique $\mathcal{C}(i)$.
   Run a combined protocol $\Pi^*(\Pi_1, \dots, \Pi_n)$ on the n protocols $\Pi_1, \dots, \Pi_n$, where
   - For each $i \in [n]$, Π_i denotes an invocation of Protocol 2 on the $|\mathcal{C}(i)| + 2$ parties $A, B, P_{j_1}, \dots, P_{j_{|\mathcal{C}(i)|}}$ with inputs (r_0, r_1) and c.[a]
6. Finally, B computes $x_b = y_b \oplus r_c$.

[a]In the case $\mathcal{C}(i) = \mathcal{C}_1$, A is both the OT sender and a member of the clique. A similar condition holds for B in the case $\mathcal{C}(i) = \mathcal{C}_2$.

that steps 3, 4 and 6 perform OT correction, that is, they perform a transformation from random OT to 1-out-of-2 OT. This yields the $n/2$-security of Protocol 5.

Proof of Theorem 3.6.1. The "only if" part of theorem has been proven by virtue of the lower bound of Theorem 3.4.2 with $t = n/2$. We now prove the "if" part. We note that in the case where A and B are in the same connected component in the network G, by the $n/2$-security of Protocol 4 and Lemma 3.5.3, we note that Protocol 5 is an $n/2$-secure OT protocol with A as the sender and B as the receiver, thus proving the theorem.

We now proceed to the case where A and B are not in the same connected component in

G. We must show that the protocol is secure against t-threshold adversaries as long as the vertices cannot be partitioned into two sets V_A, V_B each of size $t = n/2$ with $A \in V_A, B \in V_B$ such that there are no edges between V_A and V_B. Let A be a t-threshold adversary which corrupts parties T, $|T| \leq t$. We will construct a simulator S which plays the role of the uncorrupted parties.

If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so the simulator can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \varnothing$ then S chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend on the input at all are on point-to-point channels between A and B, the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that the corrupted parties T include exactly one of A, B. If $A \in T$ but $B \notin T$, then S chooses an arbitrary bit b and runs the protocol, invoking the OT simulator for each invocation of Protocol 4. It follows that as long as the combined protocol Π^* in step 5 is secure against A, Protocol 5 is secure against A. It remains to show that a majority of the n protocols $\Pi_1, \ldots, \Pi_n$ are secure against A. Since party B is honest, by Lemma 3.5.2, protocol Π_i is secure against A as long as at least one of the parties in clique $\mathcal{C}(i)$ is honest. In particular, if party P_i is honest then protocol Π_i is secure against A. At most t of the parties $P_1, \ldots, P_n$ are corrupt, so the only protocols which may be insecure against A are the t protocols Π_i corresponding to the corrupted parties P_i. Assume that all t of these protocols are insecure against A. Then the corrupted parties lie in completely corrupted cliques who sizes sum to $n/2$. This then gives a set $V_A = T$ of $n/2$ parties containing A but not B such that there are no edges from V_A to the remaining vertices $V_B = \overline{T}$. However, we know that G possesses no such partition. Hence, at most $t - 1 < n/2$ of the n protocols are insecure against A and hence by Lemma 3.5.6, the combined protocol Π^* in step 5 is secure and hence Protocol 5 is secure against A.

The remaining case that $B \in T$ but $A \notin T$ is similar. Here, the simulator S is given the output value x_b. S runs the protocol with (x_b, x_b) as the input to A, again invoking the OT simulator for each invocation of Protocol 4. As above, as long as the combined protocol

Π^* in step 5 is secure against A, Protocol 5 is secure against A. By the same argument, the only protocols Π_i which may be insecure against A are the t protocols corresponding to the corrupted parties P_i. If all t of these protocols are insecure against A, we have a set $V_A = \overline{T}$ of $n/2$ parties containing A but not B such that there are no edges from V_A to the remaining vertices $V_B = T$. However, we know that G possesses no such partition, so at most $t - 1 < n/2$ of the n protocols are insecure against A. By Lemma 3.5.6, the combined protocol Π^* in step 5 is secure and so Protocol 5 is secure against A. □

We now analyze the efficiency of Protocol 5.

Theorem 3.6.3. *Protocol 5 runs in quasi-polynomial time. Assuming one-way functions, we can obtain a computationally secure protocol which runs in polynomial time using computationally secure cascading (Lemma 3.5.4).*

Proof. Each iteration of step 1 decreases the length of a path between any pair of vertices from ℓ to $\lceil \ell + 1 \rceil / 2$. Consequently, after $O(\log n)$ iterations the graph will consist of a collection of disjoint cliques, and the protocol will move on to the next step. By Lemma 3.5.3 (Cascading), if each iteration can be performed in time at most T assuming the augmented graph, then the full cascaded protocol runs in time at most $T^{O(\log n)}$. Since $T = \mathsf{poly}(n)$ and each other step of the protocol is efficient, this implies that Protocol 5 runs in quasi-polynomial time.

Replacing the cascading of step 1 with the more efficient but computationally secure cascading of Lemma 3.5.4, we have the cascaded protocol runs in time $O(T\mathsf{poly}(\lambda) \cdot \log n)$. Since each other step of the protocol is efficient, this implies that assuming one-way functions, we have a computationally-secure version of Protocol 5 that runs in quasi-polynomial time. □

3.7 The case $t = n - 2$

On account of the lower bound proven in [HIK07], we note that $t = n - 2$ is the largest value of t for which the question is non-trivial. In this section we present an improved

Protocol 6: $n-2$ corruptions

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \dots, P_n$ be the n parties, and let graph $G = (V, E)$ be the OT network among the parties. A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. For all pairs of parties $P_i, P_j \in V$ with $i, j \geq 3$ such that $\{P_i, P_j\} \notin E$:
   (a) Invoke Protocol 5 (or any 2-secure protocol for $n' = 4$) on the induced OT subgraph $G_{i,j} := G \cap \{P_i, P_j, A, B\}$ with independent random inputs $(r_0^{i,j}, r_1^{i,j})$ and $b^{i,j}$, to obtain OT correlations along edge $\{P_i, P_j\}$.
   (b) By virtue of cascading (Lemma 3.5.3), we can add edge $\{P_i, P_j\}$ to the graph G.[a]

   The OT network G now contains a $(n-2)$-clique among vertices $P_i, \dots, P_n$.
2. Invoke Protocol 2 (t-clique) with input (x_0, x_1) and b.

[a]We will only have OT security over this edge when at least two of the parties P_i, P_j, A, B are honest, but we obtain the functionality of the edge regardless. We address security of the overall protocol in the proof.

computationally efficient OT protocol between A and B for the special case $t = n - 2$ for all $(2, A, B)$-unsplittable networks G.

Theorem 3.7.1. *Let G be an n-party OT network containing parties A and B. Then Protocol 6 is an efficient $(n-2)$-secure OT protocol between A and B if and only if G is $(2, A, B)$-unsplittable.*

The protocol is built upon the following structural aspect of the network G under consideration. Since G is $(2, A, B)$-unsplittable, for any two sets of vertices $V_A A$ and $V_B B$ such that $|V_A| = |V_B| = 2$, there exists an edge from a vertex of V_A to a vertex of V_B. In particular, this implies that for any two parties P_i, P_j where $i, j \geq 3$, the sub-network $G_{i,j}$ induced by parties A, B, P_i and P_j is $(2, A, B)$-unsplittable. Then for any i, j, we also have that the sub-network $G_{i,j}$ is $(2, P_i, P_j)$-unsplittable. Hence, we could try to obtain OT correlations between every pair of vertices P_i, P_j by running Protocol 5 on every $G_{i,j}$ for $n = 4$ parties. Notice that if these invocations were secure, then we would obtain an $(n-2)$-clique in the network after which we can execute Protocol 2 in order to obtain OT correlations between A and B. This is described in Protocol 6. However, each of the execution of Protocol 5 is only guaranteed to be secure if at most two of the corresponding parties are corrupt. This need not be true in general, and so we cannot directly leverage the security of Protocol 5. Nonetheless, we will argue that Protocol 6 is secure against $t = n-2$ corruptions.

Proof Intuition (Theorem 3.7.1): In order to analyze the $(n-2)$-security of Protocol 6, we consider each invocation of Protocol 5 on a sub-network $G_{i,j}$. If at most two of the four parties in $G_{i,j}$ are corrupt, then that invocation of Protocol 5 is secure and yields secure OT correlations between parties P_i and P_j. Appealing to Lemma 3.5.3, we can augment G to include edge $\{P_i, P_j\}$.

Each $G_{i,j}$ must contain at least one honest party since either A or B must be honest (otherwise, there is nothing to prove). It remains to consider sub-networks $G_{i,j}$ in which three of the parties are corrupt. Since at least one of A or B is honest, this implies that both P_i and P_j are corrupt. Thus, there is nothing to prove regarding the security of the invocation of Protocol 5 on $G_{i,j}$ since we are establishing OT correlations between a pair of corrupt parties P_i and P_j. Combining these claims, we have that each of the invocations of Protocol 5 is secure and yields secure OT correlations between the pairs of parties P_i, P_j for all $i, j \geq 3$. By virtue of Lemma 3.5.3, we obtain an $(n-2)$-clique in the network and the $(n-2)$-security of Protocol 2 with $t = n-2$ proves the $(n-2)$-security of Protocol 6.

Proof. The "only if" part of theorem follows from the lower bound of Theorem 3.4.2 with $t = n-2$. We now prove the "if" part. Let A be a t-threshold adversary which corrupts parties T, $|T| \leq t = n-2$. If A and B are both corrupt then the uncorrupted parties receive no input, so the simulator S can perfectly simulate the uncorrupted parties.

Otherwise we have that at least one of A or B is uncorrupted. We first show how to simulate step 1. Since G is $(2, A, B)$-unsplittable, for each pair of parties P_i, P_j considered in step 1, we have that $G_{i,j}$ is $(2, A, B)$-unsplittable. Since $\{P_i, P_j\} \notin E$, this implies that some vertex among A, B, P_i, and P_j has degree at least 2 in $G_{i,j}$. In particular, we have $G_{i,j} \not\sqsubseteq_{P_i,P_j} \Lambda_2^0$, where Λ_2^0 is labeled so that P_i and P_j are in separate components. Consequently, by Theorem 3.6.1, we securely obtain OT correlations between P_i and P_j whenever at most two of the parties A, B, P_i, P_j are corrupt.

Since A and B are not both corrupt, if P_i and P_j are also not both corrupt, then at most two of the four parties A, B, P_i, P_j are corrupt. Consequently we securely obtain OT correlations between P_i and P_j. Otherwise, P_i and P_j are both corrupt, in which case there is nothing to prove. More formally, there is a simulator which can perfectly simulate the invocation of Protocol 5 in step 1a by executing the protocol of the honest parties, since the honest parties receive no input. Consequently we securely obtain OT correlations between the corrupted parties P_i and P_j. In both cases, by Lemma 3.5.3, we have an OT simulator which can simulate subsequent invocations of the OT channel between P_i and P_j.

Therefore, by the end of step 1 we t-securely obtained an OT channel between every pair of parties P_i, P_j for $i, j \geq 3$. Then by Lemma 3.5.2, the invocation of Protocol 2 in step 2 is a t-secure OT protocol with A as the sender and B as the receiver, as desired. □

3.8 The General Case: $t \geq n/2$

In this section, we resolve the network OT question for general $t \geq n/2$. Note that from the protocols in Sections 3.6 and 3.7 we already have tight answers for the special cases $t = n/2$ and $t = n - 2$. We address the general question from both ends of the spectrum, namely for t larger than $n/2$ and t smaller than $n - 2$. These analyses yield two distinct protocols which employ the protocols from Sections 3.6 and 3.7 as their respective base cases. The two protocols we describe are efficient in different parameter regimes. Protocol 7 described in Section 3.8.1 is quasi-polynomially efficient[13] when $t = n/2 + O(1)$, and Protocol 8 described in Section 3.8.2 is (polynomially) efficient when $t = n - O(1)$. Putting these protocols together, we obtain a single protocol that is efficient under computational security when either $t = n/2 + O(1)$ or $t = n - O(1)$. We note that the problem of recognizing whether there exists a t-secure OT protocol is efficient in these cases, while the recognition problem for general n, t is coNP-complete.

[13]or polynomially efficient under computational security

3.8.1 General Protocol (Quasi-polynomial for $t = n/2 + O(1)$)

We now describe a t-secure OT protocol between A and B for all $(n-t, A, B)$-unsplittable networks G. As a consequence of the lower bound described in Section 3.4, this result is tight.

Theorem 3.8.1. *Let G be an n-party OT network containing parties A and B, and let $t \geq n/2$. Then Protocol 7 is a t-secure OT protocol between A and B if and only if G is $(n-t, A, B)$-unsplittable. The protocol achieves perfect security and runs in quasi-polynomial time for $t = n/2 + O(1)$. Assuming one-way functions, we can also obtain a protocol which achieves computational security and runs in polynomial time for $t = n/2 + O(1)$.*

The protocol proceeds by recursion, reducing the problem of obtaining an OT protocol on an n-vertex graph with $t > n/2$ corrupted parties to a number of instances of n'-vertex graphs, a majority of which have at most t' corrupted parties, for $n' = n - 1$ and $t' = t - 1$. As shown below, each n'-vertex subgraph G' has a structure similar to G in the sense that G' is $(n' - t', A, B)$-unsplittable whenever G is $(n - t, A, B)$-unsplittable. We can now recurse on these smaller problem instances, invoking an OT combiner to obtain the full protocol.

More precisely, the protocol constructs $n - 2$ subgraphs on $n - 1$ vertices, where each subgraph is obtained by deleting a single vertex other than A and B. We can recursively run a $(t - 1)$-secure OT protocol on each of the subgraphs. The final protocol invokes a combiner on these $n - 2$ candidate protocols. It remains to be shown that a majority of the subgraphs G' contain at most $t - 1$ corrupt parties.

Proof Intuition (Theorem 3.8.1): We may assume that at least one of A or B is honest. As described above, we wish to argue that a majority of the subgraphs G' contain at most $t - 1$ corrupt parties. Combining this with the claim that these subgraphs preserve an unsplittability property of G and invoking Lemma 3.5.6 completes the proof.

However, this claim follows from the following observation. Since $t > n/2$, if exactly t parties are corrupt then a majority of the subgraphs contain at most $t - 1$ corrupt parties since A and B are not both corrupt. If strictly fewer than t parties are corrupt then all

Protocol 7: General Protocol I

Preliminaries: Let $A, B, P_3, \ldots, P_n$ be the n parties in a network G and let $t \geq n/2$ be the maximum number of corruptions. A has input (x_0, x_1), and B has input $b \in \{0,1\}$.

Protocol:

1. If $t = n/2$, then invoke Protocol 5 and halt.
2. Otherwise, run a combined protocol $\Pi^*(\Pi_3, \ldots, \Pi_n)$, where
   - For each $i \geq 3$, Π_i denotes the recursive invocation of this protocol on the $n-1$ parties excluding party P_i with the induced sub-network $G \setminus \{P_i\}$ and $t' = t-1$ corruptions.

of the sub-graphs contain at most $t-1$ corrupt parties. In either case, for a majority of subgraphs, at most $t-1$ of the parties are corrupt.

We first present and prove a structure lemma.

Lemma 3.8.2. *Given graph $G = (V, E)$ and a vertex i, let G_i be the induced graph on the $n-1$ vertices $V \backslash \{i\}$. If G is $(n-t, A, B)$-unsplittable, then G_i is also $(n-t, A, B)$-unsplittable.*

Proof. We will prove the contrapositive. Suppose that $G_i \subseteq_{A,B} \Lambda_{n-t}^{2t-n-1}$. This means there exists a partition of the vertex set of G_i as $V \setminus \{i\} = V_A \dot{\bigcup} V_S \dot{\bigcup} V_B$ with no edges between V_A and V_B, where $A \in V_A$, $B \in V_B$, $|V_A| = |V_B| = n-t$ and $|V_S| = 2t-n-1$. But then we can partition the vertex set of G as $V = V_A \dot{\bigcup} V'_S \dot{\bigcup} V_B$, where $V'_S = V_S \cup \{i\}$. We have that $|V_A| = |V_B| = n-t$ and $|V'_S| = 2t-n$, and there are no edges between V_A and V_B, so $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$, which is a contradiction. □

Using the lemma, we now prove Theorem 3.8.1.

Proof of Theorem 3.8.1: The "only if" direction follows from the lower bound of Theorem 3.4.2, and the efficiency claim follows immediately from Theorem 3.6.3. We now prove the "if" direction by induction on $2t-n$. The base case of $2t-n=0$ follows from Theorem 3.6.1. Now assume for induction that the statement holds for $n' = n-1, t' = t-1$ with $t > n/2$.

Let A be a t-threshold adversary which corrupts parties T, $|T| \leq t$. If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so the simulator can perfectly simulate the uncorrupted parties. Consequently it suffices to consider the case $\{A, B\} \cap T \leq 1$. By Lemma 3.8.2, each sub-network $G \backslash \{P_i\}$ is $(n'-t', A, B)$-unsplittable. For each $i \geq 3$, let $T_i = T \backslash \{P_i\}$ denote the corrupted parties in sub-network $G \setminus \{P_i\}$. If $|T| < t$ then $|T_i| \leq |T| \leq t'$ for all i. Otherwise, $|T| = t$. In this case, since $\{A, B\} \cap T \leq 1$, it follows that at least $t-1$ of the $n-2$ parties $P_3, \ldots, P_n$ are corrupted. Since $t > n/2$, this is a majority of the parties $P_3, \ldots, P_n$. For each corrupted party P_i, $|T_i| = t'$. Consequently in each case we have that a majority of the sets $T_3, \ldots, T_n$ satisfy $|T_i| \leq t'$. Therefore by our inductive assumption, a majority of the protocols $\Pi_3, \ldots, \Pi_n$ in step 2 are secure against A. By Lemma 3.5.6, we have that the combined protocol Π^* is secure against A, so by induction, the theorem holds for all n, t with $t \geq n/2$. $\square$

As an immediate consequence, the condition described in Theorem 3.8.1 is both necessary and sufficient in order to obtain a complete network of OT channels and perform secure multiparty computation among all parties in the network.

Corollary 3.8.3. *Let G be an n-party network. For $t \geq n/2$, we can t-securely generate OT correlations between all pairs of parties (thus, completing the OT network) if and only if the G is $(n-t)$-unsplittable.*

Proof. If G is $(n-t)$-unsplittable, then for all pairs of vertices A, B, G is $(n-t, A, B)$-unsplittable and hence from Theorem 3.8.1, we can generate OT correlations between A and B. Conversely, if $G \subseteq \Lambda_n^{2t-n}$, then choosing vertics $A \in V_A$ and $B \in V_B$, we have that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$ and hence Theorem 3.8.1 rules out the existence of such a protocol. Hence, there exist a pair of vertices between whom we cannot t-securely generate OT correlations. This completes the proof. $\square$

3.8.2 General Protocol (Efficient for $t = n - O(1)$)

We now describe another t-secure OT protocol for all networks G with A as the sender and B as the receiver whenever the network G is $(n-t, A, B)$-unsplittable. This protocol uses,

Protocol 8: General protocol II

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \ldots, P_n$ be the n parties in a network $G = (V, E)$. A has input (x_0, x_1), and B has input $b \in \{0, 1\}$. Let $k = n - t$.

Protocol:

1. Invoke Protocol 6 with $t' = n - 2$ on the n'-node network G' with inputs (x_0, x_1) and b, where $n' = \binom{n-2}{k-1} + 2$, and
   - S_{k-1} is the set of subsets of $\{P_3, \ldots, P_n\}$ of size $k - 1$.
   - The n' vertices of G' correspond to A, B, and the $\binom{n-2}{k-1}$ subsets of S_{k-1}.
   - The edges of G' are defined as follows. Two subsets $X, Y \in S_{k-1}$ will have an edge if either $X \cap Y \neq \varnothing$ or there exists a pair of parties $P_i \in X$ and $P_j \in Y$ with $\{P_i, P_j\} \in E$.
   - Invocation of OT over an edge $\{X, Y\}$ in G' with inputs (z_0, z_1) and c is performed as follows.
     - If $X \cap Y \neq \varnothing$, then choose some party $P_i \in X \cap Y$. $P_i \in X$ and hence knows (z_0, z_1); similarly, $P_i \in Y$ and knows c. Consequently P_i knows z_c, and sends it to the other members of set Y.
     - If $X \cap Y = \varnothing$, there is a pair of parties $P_i \in X, P_j \in Y$ such that $\{P_i, P_j\} \in E$. P_i knows (z_0, z_1) and P_j knows c, so they can invoke OT over the channel (P_i, P_j) in G, and P_j can then send the value z_c to the other members of set Y.

in spirit, a reduction in the opposite sense than the one described in Section 3.8.1. The protocol is efficient whenever $t = n - O(1)$.

Theorem 3.8.4. *Let G be an n-party OT network containing parties A and B, and let $t \geq n/2$. Protocol 8 is a t-secure OT protocol between A and B if and only if G is $(n-t, A, B)$-unsplittable. The protocol is efficient for $t = n - O(1)$.*

The idea behind this protocol is the following. We increase the size of the network in order to obtain a large number N of well-connected additional simulated parties such that at least one them is guaranteed to be honest. We may assume that at least one of A and B is honest, as otherwise there is nothing to prove. Consequently there are at least two honest parties in the augmented network. We will now apply the protocol from Section 3.7. It remains to describe the construction of these simulated parties, to show that at least one of them is honest, and to prove a structural lemma that if the original network G is $(n - t, A, B)$-unsplittable then the augmented network G' is $(2, A, B)$-unsplittable.

Proof Intuition (Theorem 3.8.4): We first describe the new network generated by Protocol 8. The parties other than A and B in the newly constructed network consist of all subsets of size $n-t-1$ of the parties in G containing neither A nor B. Lemma 3.8.5 below shows that this new network G' is $(2, A, B)$-unsplittable whenever G is $(n-t, A, B)$-unsplittable, where the edges of G' are as described in Protocol 8. A party X in G' will be considered honest if all constituent parties $P_i \in X$ from G are honest. Since one of A and B is honest and at most t parties are corrupt, at least $n-t$ parties are honest and in particular, at least $n-t-1$ of the parties other than A and B must be honest. This means that one of the subsets is completely honest. Since A or B is also honest, G' is guaranteed to have at least two honest parties. Combining these facts and invoking Theorem 3.7.1 completes the argument.

We will use the following structural lemma about the network G' constructed in Protocol 8.

Lemma 3.8.5. *If G is $(n-t, A, B)$-unsplittable, then G' is a $(2, A, B)$-unsplittable network on $n' = \binom{n-2}{n-t-1} + 2$ vertices, where G' is the network produced in Protocol 8.*

Proof. We prove the contrapositive. Assume that $G' \subseteq_{A,B} \Lambda_2^{n'-2}$. Let $k = n-t$, and for $i \in \mathbb{N}$, let S_i denote the set of subsets of $V \setminus \{A, B\} = \{P_3, \ldots, P_n\}$ of size i. Then there exist vertices $X, Y \in S_{k-1}$ such that there are no edges in G' between any of the parties in $\{A, X\}$ and any of the parties in $\{B, Y\}$. In particular, $X \cap Y = \varnothing$, since otherwise $\{X, Y\}$ would be an edge of G'. This implies that we have $2k = 2(n-t)$ parties $\{A, B\} \cup X \cup Y$ such that there are no edges in G from the $n-t$ parties $\{A\} \cup X$ to any of the $n-t$ parties $\{B\} \cup Y$. By definition, this means that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$, which is a contradiction. $\square$

Using this lemma, we will now prove the correctness of Protocol 8.

Proof of Theorem 3.8.4. As before, we only need to prove the "if" part. Let A be a t-threshold adversary which corrupts parties T, $|T| \leq t$. If A and B are both corrupt, then the honest parties have no input, so the simulator S can perfectly simulate the uncorrupted parties. If A and B are both honest, then S chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend at all on the inputs are on point-to-point channels between A and B, the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that at most $t-1$ of the parties $P_3, \dots, P_n$ are corrupt and that either A or B is honest. In particular, for $k = n - t$, there are at least $k - 1$ uncorrupted parties among $P_3, \dots, P_n$. Consequently, S_{k-1} contains some set X consisting only of honest parties. We treat a party X in G' as honest if all constituent parties $P_i \in X$ are honest. Since A and B are not both corrupt, we have that G' contains at least two honest parties. By Lemma 3.8.5, G' is $(2, A, B)$-unsplittable. Consequently by Theorem 3.7.1 there is a simulator S' which simulates the roles of the honest parties in the invocation of Protocol 6 on G' in step 1. We define a simulator S for Protocol 8 which behaves exactly as an honest party for communication within each party $X \in S_{k-1}$ and invokes S' for any communication between parties in G'. The behavior of this protocol is identical to the behavior of S'. Hence, by the correctness of simulator S', we have that the view of the adversary is identical in the real and ideal worlds. □

3.9 Bounding the number of edges in $\approx \frac{n}{2}$-unsplittable graphs.

In this section we discuss the minimum number of edges in a graph which is $(n - t)$-unsplittable for $t = \lfloor (n + 1)/2 \rfloor$. We show that the minimum edge count is $n/2$ if n is even and $t = n/2$, and $(n + 3)/2$ if n is odd and $t = (n + 1)/2$. These bounds give the minimum number of OT channels required to obtain t-secure MPC among n parties in a network for $t = \lfloor (n+1)/2 \rfloor$. They constitute the first nontrivial cases, since no OT channels are needed in the case of an honest majority.

Theorem 3.9.1. *Let n be even and $t = n/2$. Then any $(n - t)$-unsplittable graph must contain at least t edges. This bound is tight.*

Proof. To show that the bound is tight, note that the t-claw graph (Figure 3-4a) (or any graph with a tree on $t + 1$ vertices and no other edges) has t edges and contains a connected component consisting of $t+1 = n/2+1$ vertices, so it is $(n/2)$-unsplittable. We now show that every graph with fewer edges can be split.

Let G be a graph containing $t-1$ edges. Let m be the number of connected components of G, let $C_1, C_2, \ldots, C_m$ be the components in nonincreasing order of size, and let $\alpha_i = |C_i|$, so that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$ and $\sum_{i=1}^m \alpha_i = n$. G is $(n-t)$-splittable if and only if there is some subset $I \subset [n]$ such that $\sum_{i \in I} \alpha_i = n/2$, that is, some subset of the values α_i sum to $n/2$.

For all natural numbers x, let $a_x = |\{i : \alpha_i \geq x\}|$ denote the number of *connected components* with size at least x, and let $b_x = |\{v : v \in C_i \wedge \alpha_i < x\}|$ denote the number of *vertices* in connected components with size smaller than x. Note that a_x counts the components of certain sizes, while b_x counts the vertices contained in components of certain sizes. Since a component of size s must contain at least $s-1$ edges, for any $x > 1$ we have that $a_x \leq (t-1)/(x-1) < n/(2x-2)$. In particular, we have that $a_2 < n/2$, $a_3 < n/4$, and $a_4 < n/6$.

Since G has at most $n/2-1$ edges, at least two vertices have degree zero, so $\alpha_1 = \alpha_2 = 1$. Consequently $b_2 \geq 2$. A set of c components containing at most $t-1$ edges can include at most $c+t-1$ vertices. Consequently, for any $x > 1$, we have that

$$b_x \geq n - (a_x + t - 1) = \frac{n}{2} + 1 - a_x.$$

In particular, we have that $b_3 > \frac{n}{4} + 1$ and $b_4 > \frac{n}{3} + 1$.

Since $\alpha_1 \geq \alpha_2 \geq \alpha_3$, the third largest component has size $\alpha_3 \leq n/3$. There are only $n/2-1$ edges, so $\alpha_i \leq n/2$ for all i.

Consider the following greedy algorithm. Initially let $S_2 = C_2$. For i from 3 to n, if $|S_{i-1} \cup C_i| \leq n/2$ then set $S_i = S_{i-1} \cup C_i$, and otherwise set $S_i = S_{i-1}$. We show that at the end of this loop, the set S_m will always have size $n/2$.

Since $|C_m| \leq n/2$, the sum of the sizes of the other components is at least $n/2$, so if $|S_{i+1} \cup C_i| \leq n/2$ for every i considered in the loop then the loop must terminate with $|S_1| = n/2$. Otherwise there is some i such that $|S_{i-1} \cup C_i| > n/2$. Choose the last iteration

i for which this is true. Since $i \geq 3$, we must have that $\alpha_i \leq \alpha_3 \leq n/3$. If $\alpha_i > 3$, then since $n/2 - |S_i| \leq \alpha_i - 1 \leq n/3 - 1$ and $b_4 > \frac{n}{3} + 1$, we reach a contradiction with the assumption that i is the last such iteration, since there are enough vertices in components $C_{i+1}, \ldots, C_m$ to make some subsequent $|S_{j-1} \cup C_j| > n/2$. Otherwise $\alpha_i \leq 3$, so since $n/2 - |S_i| \leq \alpha_i - 1 \leq 2$, so since $b_2 \geq 2$, we must have that $|S_m| = n/2$.

Consequently at the end of the loop we always have that $|S_m| = n/2$. Since S_m consists only of entire connected components, S_m consists of $n/2$ vertices with no edges to the rest of the graph, so $G \subset \Lambda^0_{n/2}$ cannot be $(n/2)$-unsplittable. □

Theorem 3.9.2. *Let n be odd and $t = (n+1)/2$. Then any $(n-t)$-unsplittable graph must contain at least $t+1$ edges. This bound is tight.*

Proof. To show that the bound is tight, note that the $(t+1)$-edge graph consisting of a cycle on $t+1$ vertices is $(n-t)$-unsplittable, since removing any vertex still leaves a connected component of size $t-1$. We now show that every graph with fewer edges can be split.

Let $G = (V, E)$ be a graph containing $|E| = t$ edges. If the maximum degree is 1, then G contains at most $n/2 < t$ edges, a contradiction. Let v be a vertex of maximal degree ≥ 2. The induced subgraph on vertex set $V \setminus \{v\}$ contains $n-1$ vertices and at most $t-2 < (n-1)/2$ edges. By Theorem 3.9.1 it cannot be $((n-1)/2)$-unsplittable, so G cannot be $((n-1)/2)$-unsplittable either. □

Chapter 4

Repudiability and claimability of ring signatures

This chapter is based on joint work with Sunoo Park [PS19].

4.1 Introduction

Ring signatures, introduced by [RST01], 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 called a "ring." Ring signatures can be useful, for example, to certify that certain leaked information comes from a privileged set of government or company officials without revealing the identity of the whistleblower, to issue important orders or directives without setting up the signer to be a scapegoat for repercussions,[1] or to enable untraceable transactions in cryptocurrencies (as in Monero [Mon]).

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 the verification key. Signers can produce signatures that verify with respect to any set of verification keys that includes

[1]When it comes to national security issues, for instance, there may be some reluctance among law-makers to "roll back" existing laws or reduce checking or surveillance measures, which could be due in part to the risk of ending up a scapegoat upon any future national security incident like a terrorist attack.

their own, and unforgeability guarantees that no party can produce a valid signature with respect to a set of verification keys without possessing a corresponding secret key.

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 nor to claim a signature in a convincing way. However, depending on the threat model, we believe that the opposite guarantees—that is, to guarantee the ability to denounce or claim signatures—may be useful too, as elaborated below. Furthermore, whatever one's preference, a guarantee one way or the other seems more desirable than no guarantee either way.

Prior security definitions for ring signatures do not conclusively provide these guarantees one way or the other. That is, a non-signer might be able to *repudiate* a signature that he did not produce ("repudiability"), or this might be impossible ("unrepudiability"). Similarly, a signer might be able to later convincingly claim that a signature he produced is indeed his own ("claimability"), or be unable to do so ("unclaimability").

The most detailed taxonomy of security definitions for ring signatures was given by [BKM09], which presents a series of anonymity guarantees of increasing strength. A natural anonymity guarantee defined by [BKM09], called "anonymity against adversarially chosen keys," is informally described as follows: an adversary who controls all but $t \geq 2$ 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 t honest parties produced a given signature. This anonymity definition might allow a party to ascertain whether a given signature was produced by her own signing key, and perhaps also to convince others of this fact—but it does not *guarantee* or *prohibit* either of these capabilities.

On the other hand, the strongest of the anonymity definitions of [BKM09] (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 herself cannot tell the difference between a signature produced using her own signing key and one produced using someone else's.

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 desirable to prevent distinguishing one's own signatures from those generated by someone else. On the other hand, without the ability to distinguish, it would be virtually impossible to tell if someone had stolen your signing key. Moreover, as discussed below, it could be beneficial in certain circumstances for members of a ring to have the ability to disown signatures of messages that they have strong reasons to denounce; and conversely, in some circumstances the signer of a message might later wish to prove to the world that he was the one who produced a particular signature in the past.

We have now identified four potentially useful notions for ring signatures: *repudiability*, *unrepudiability*, *claimability*, and *unclaimability*. The main contributions in this paper consist both of new definitions and constructions of each of these notions. Before diving into an overview of definition and constructions, we provide some discussion of why each of these notions—some of which directly oppose each other—may be meaningful and desirable: the following scenarios explore a few of the circumstances in which various of the above guarantees might be appropriate. Though some of the scenarios are phrased somewhat whimsically, we believe that each scenario illustrates a meaningful threat model motivating the definition concerned.

Scenario 1 (Repudiability) 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 tendency to prefer whomever appears friendlier in a series of nationally televised grinning 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 allay the doubt in the minds of her suspicious detractors. As mentioned above, ring signatures are deliberately designed to allow *anyone* to attach anyone else's name to a signature, without the latter's knowledge or consent. Despite this, there could be realistic situations in which non-signing members of a ring associated with a particular message could suffer serious consequences through no fault of their own, perhaps due to the real signer adversarially trying to damage their reputation. In light of this, perhaps it would be desirable in some contexts for the owner of a verification key to be able to denounce messages, e.g., to clear her name of a crime or hate speech accusation that might otherwise impact her life in terms of reputation, job prospects, or incarceration.

Scenario 2 (Claimability) Our next story concerns a talented brewery employee who developed new statistical techniques to test the quality of beers. Naturally, his employer was protective of its competitive advantage since other breweries at the time may not have been using similar statistical methods. Yet, in the interest of science, they allowed him to publish his results—on condition of anonymity.[2] A credible way to prove authorship at a later date, after the need for anonymity has ceased to exist, might be very useful—especially in case of competing claims by impostors. As we see here, claiming authorship of an anonymous work may become appropriate after a passage of time. The next example illustrates quite a different type of situation in which claimability at the signer's discretion may be valuable.

Consider an employee Emily who is concerned about unethical practices at her company, and takes it upon herself to expose what is going on and publish a critical commentary.

[2]This is the true story of William Sealy Gosset's invention of the Student's t-test at Guinness Brewery in 1908 [Man00].

Concerned about her job security and possible retribution, as well as the credibility of her allegations, she maintains her anonymity using ring signatures. It emerges, in fact, that similar practices are prevalent across the industry: related revelations drive a wider movement of reform. Some time later, after her company has substantially reformed its practices and her fears of retribution have been allayed—perhaps by her promotion, or by a change in leadership—Emily seeks to reveal her identity and add her voice to the growing movement, providing her solidarity, legitimation, and follow-up story. In addition, if following the reforms, those involved in the earlier unethical practices were subject to stigma or even prosecution, claimability of her earlier ring signatures would allow Emily to exculpate herself.

Scenario 3 (Unrepudiability and unclaimability) Let us return to the government of the fictional country of Oz. The parliament of Oz is mired in partisan gridlock, with legislators from each party ruthlessly voting down any bills, however reasonable, proposed by members of the opposing party—preventing any laws at all from being enacted and effectively shutting down the government, which is in no party's interest. Suppose that instead of directly proposing a new law, a legislator of Oz anonymously publishes the text of the proposed bill using a ring signature scheme:

Proposed: that free ice cream shall be provided every Tuesday.[3]
Signed: a member of the Parliament of Oz.

If the signer used an unclaimable ring signature scheme, then she could not decide to reveal her identity upon a later change of heart, allowing legislators of both parties to support or oppose the bill on its merits without worrying about purely political considerations.

Unclaimability and unrepudiability may be particularly useful guarantees in scenarios where the placement of whole groups of people under duress is a substantial concern. For instance, in circumstances where an employer or authoritarian government may coercively compel individuals to provide a repudiation or proof of authorship (e.g. signing randomness)

[3]Even if each party might support this legislation, they may be unwilling to do so if it were proposed by the other party, decrying their respective opponents as either fiscally irresponsible or in the pocket of Big Ice Cream.

for a signature, the provable inability to do so convincingly may be essential. Unrepudiability may also be desirable in situations in which members of a ring are likely to have conflicting individual incentives but there is a possibility of collective benefit in case of cooperation, as in a prisoner's dilemma scenario.

Summary of technical contributions. We formalize *repudiability*, *unrepudiability*, *claimability*, and *unclaimability* of ring signatures, as well as strengthened anonymity and unforgeability definitions which are compatible with each of these notions. We show that *unclaimability* implies *unrepudiability* (intuitively, because a failed repudiation can be used as a claim). Anonymity against adversarially chosen keys is the strongest anonymity notion compatible with *repudiability* and *claimability*, and anonymity against full key exposure is implied by *unclaimability* and equivalent to *unrepudiability*.

We provide three constructions based on different assumptions, one for each of the three notions of *repudiability*, *claimability*, and *unclaimability*. Perhaps the most surprising of these is *unclaimability*, which guarantees that the signer cannot later credibly convince others that she produced a particular signature. 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 in the ring can also produce credible signing randomness* for any signature in which they are implicated. Our construction of *unclaimable* ring signatures is an augmentation of the lattice-based ring signature scheme of [BK10] that adds additional algorithms allowing anyone in the ring to generate credible signing randomness; this capability is achieved via lattice trapdoors.

Our construction of *repudiable* ring signatures is based on verifiable random functions (VRFs), which are implied by either the (strong) RSA assumption, assumptions on bilinear maps, or NIWIs and commitments; see [Bit17, GHKW17] and references therein for more detailed discussion of the assumptions that imply VRFs.[4] Our construction does not use

[4]Note that the ring signature construction of [BKM09] assumes the closely related assumption of NIZKs,

standard ring signatures as a building block, and as such can also be viewed as a new construction of standard ring signatures. Our construction of *claimable* ring signatures, on the other hand, is a simple and generic black-box transformation from any standard ring signature scheme to a claimable one. We overview our contributions in more detail below.

4.1.1 Definitional contributions

Repudiability. We define a *repudiable ring signature scheme* as a ring signature scheme that is equipped with additional algorithms Repudiate and VerRepud as follows. Repudiate takes as input a signing key sk, a ring signature σ, and a "ring" R (i.e., a set of verification keys), and outputs a *repudiation* ξ. VerRepud takes as input a ring R, a signature σ, a repudiation ξ, and a verification key vk, and outputs a a single bit indicating whether or not ξ is a valid repudiation attesting that σ 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: we need the release of repudiations not to compromise the unforgeability or anonymity of any future signatures. Accordingly, we modify the definitions of unforgeability and anonymity for repudiable ring signatures (Definitions 4.3.4 and 4.3.5), by additionally giving the adversary access to a *repudiation oracle*. This ensures that repudiations of past

which are implied by VRFs and also imply VRFs if the NIZK is subexponentially hard under a standard derandomization assumption; see [Bit17] for details.

signatures do not affect the security guarantees of future signatures. See Section 4.3.1 for formal definitions of repudiability.

Claimability. We define a *claimable ring signature scheme* as a ring signature scheme equipped with additional algorithms Claim and VerClaim as follows. Claim takes as input a signing key sk, a signature σ, and a ring R, and outputs a claim ζ. VerClaim takes a input a ring R, a verification key vk, a signature σ, and a claim ζ, and outputs a single bit indicating whether or not ζ is a valid claim attesting that σ was produced by vk. The three requirements for a claimable ring signature scheme are, informally, as follows.

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.

3. *No framing:* No adversary can produce a signature together with a valid claim of that signature on behalf of an honest (non-signing) party.

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 4.3.10 and 4.3.11), by additionally giving the adversary access to a *claim oracle.* See Section 4.3.3 for formal definitions of repudiability.

Repudiability and *claimability* are compatible, i.e., a ring signature scheme can be both repudiable and claimable. Indeed, our repudiable and claimable constructions together give rise to such a scheme. Notably, the unforgeability and anonymity definitions corresponding to the natural notion of 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 a stronger adversary which is simultaneously given access to both a repudiation oracle and a claim oracle. See Section 4.3.5 for further discussion on repudiable-and-claimable schemes.

Unclaimability We also introduce *unclaimable* ring signature schemes, in which the signer *provably cannot* convincingly claim that she was the one who produced the signature. As briefly mentioned above, while the signer can always save the signing randomness and reveal it along with her secret key in an attempt to claim authorship of a signature, it is not always true that this constitutes a convincing claim. In particular, such a claim is not credible if *any* member of the ring can take a valid signature and produce fake randomness that produces the desired signature using her own signing key.

The idea that a non-signer can adaptively produce fake randomness is reminiscent of deniable encryption [CDNO97], in which an encryptor and/or recipient is required to produce fake randomness "explaining" that a particular ciphertext is an encryption of an adversarially chosen message.

We define an *unclaimable* ring signature scheme to capture just this requirement: that is, any member of the ring must be able to produce fake signing randomness for a signature that is distributed indistinguishably from real signing randomness. Intuitively, the only information potentially possessed by a signer but not by the other members of the ring is the signing randomness, so non-signers that can generate convincing simulated signing randomness can also convincingly simulate any additional information that might be released by the signer in an attempt to claim the signature. We consider a strong flavor of this definition in which the indistinguishability property, described informally below, is statistical.

1. *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 corresponding signing randomness used by that individual to sign the same message, even given all verification keys and signing keys.

We formally define unclaimability in Section 4.3.4.

Remark 1. 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.

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 that 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. Unrepudiability is equivalent to anonymity against full key exposure and is implied by unclaimability.

Repudiable	VRF (Section 4.4)
Unrepudiable	RS anonymous against FKE (Section 4.3.2)
Claimable	Transformation from any RS (Section 4.5)
Unclaimable	SIS (Section 4.6)

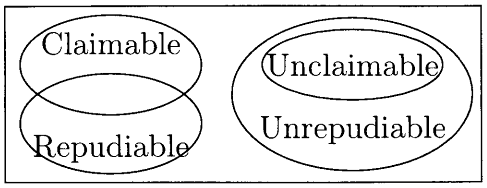


Figure 4-1: Summary of our results and assumptions relied on. VRF = verifiable random function, RS = ring signature, FKE = full key exposure, SIS = short integer solution problem.

4.1.2 Overview of our constructions

Our repudiable construction Our construction relies on ZAPs (two-round public-coin witness-indistinguishable proofs) and verifiable random functions (VRFs) as building blocks.[5] Our building blocks have some overlap with those of the ring signature construction of [BKM09], which uses ZAPs, public-key encryption (PKE), 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 PKE or signature schemes, and instead use VRFs directly to achieve unforgeability and repudiability. The structure of our construction is thus very different from that of [BKM09].

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 overcome this difficulty by relying on the witness indistinguishability property of the ZAP and ensuring that the repudiation does not

[5]VRFs imply ZAPs, so it suffices to assume VRFs. [GO92, DN07b]

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 standard ring signatures based on VRFs.)

Our claimable construction We give a generic transformation from any standard ring signature scheme RS to a claimable one. The transformation uses commitment schemes, standard signatures, and PRFs (which are all achievable from one-way functions). The basic idea is to take a signature σ_{RS} under RS and append to it a *commitment* c to $(vk, \sigma_{\mathsf{RS}})$ where vk is the verification key of the signer. The verification algorithm simply checks whether σ_{RS} verifies. The claim consists of a decommitment revealing that c is a commitment to $(vk, \sigma_{\mathsf{RS}})$. 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 $(\sigma_{\mathsf{RS}}, vk')$ where vk' is not his own key but that of someone else in the ring? This ability would violate equation (4.6) of Definition 4.3.9 (claimability). To prevent such behavior, our construction actually commits to a *standard (non-ring) signature* on $(vk, \sigma_{\mathsf{RS}})$. 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 issue 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 not be stateful between signing and claiming; indeed, Definition 4.3.9 requires this. 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.

Remark 2. Among the constructions presented in this paper, claimability is by far the simplest. Moreover, as a generic transformation, it has the advantage of adding minimal efficiency overhead to the existing state of the art in ring signatures. The simplicity of achieving claimability is perhaps unsurprising in light of the natural intuition that claiming should be possible simply by remembering the signing randomness. As evidenced by *un*claimability, this intuition is not strictly true in general, as in certain schemes, producing signing randomness may not prove authorship. In a nutshell, our generic transformation ensures that signing randomness is indeed a convincing proof of authorship in the resulting scheme, and moreover builds into the scheme a simple method of efficiently recovering the signing randomness without storing it explicitly.

Our unclaimable construction Our construction of unclaimable ring signatures is an extension of the SIS-based ring signature scheme of Brakerski and Kalai [BK10]. 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 4.6. 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

$$\sum_i A_i x_i = y,$$

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 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. [BK10] first describes 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 4.6, 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.

Remark 3. The idea that a non-signer of a given signature can adaptively produce fake signing randomness is reminiscent of deniable encryption [CDNO97], in which an encryptor of a given ciphertext can adaptively produce fake randomness consistent with it being an encryption of a different message. In this context, it may seem somewhat surprising that our construction relies on a relatively standard assumption (SIS) while many natural definitions of deniable encryption are not known to be achievable without heavier assumptions such as indistinguishability obfuscation [SW14, CPP18]. A subtle difference that is significant here is that a deniably encrypted message must still be recoverable by the honest decryptor, while in the unclaimable ring signature setting, the signer's identity need not be recoverable by anyone.

4.1.3 Other related work

Several constructions of ring signatures based on lattice assumptions have been proposed (e.g., [BK10, MBB$^+$13, BLO18]). The only other construction of ring signatures based on

ZAPs is [BKM09], 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., [Ngu05, SS10, BCC$^+$15]).

Two additional works in the lattice trapdoor literature bear mentioning: the seminal [Ajt99], and the more recent [MP12]. The latter is more recent than [GPV08], whose trapdoors our unclaimable construction relies on (this reliance is carried over from the [BK10] construction).

Ring signatures with additional guarantees Since the original proposal of ring signatures by [RST01], various variant definitions have been proposed. For example, *linkable* ring signatures [LWW04] allow 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* [LSW06] 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 [FS07] considers a setting where signatures are generated with respect to "tags" and each member may sign at most a single message (say, a vote) with respect to a particular tag, or else his identity will be revealed. *Accountable* ring signatures [XY04, BCC$^+$15] allow a signer to assign the power to de-anonymize her signature to a specific publicly identified party.

It may seem that some of these variants of ring signature schemes have properties that would be useful for constructing *claimable* ring signatures as introduced in this paper. This implication is unsurprising in the context of our results: *all* of the above types of ring signature schemes in fact imply claimable ring signatures, since our construction of claimable ring signatures is a generic transformation from *any* ring signature scheme. It is unclear if leveraging the additional features of variant schemes would be more desirable than applying our generic transformation, which has very low overhead and moreover can be applied to a simpler, more efficient ring signature scheme that may lack these additional properties.

Group signatures Group signatures [CvH91] are a different type of signature that 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, described above, have been applied to the group signature setting as well. Notably, there has also been proposed a notion of *deniable group signatures* [IEH$^+$16], 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. [LNWX17] 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.

4.2 Anonymity and unforgeability of ring signatures

This section overviews standard ring signature definitions: syntax, correctness, anonymity, and unforgeability. We express the anonymity and unforgeability definitions differently from prior work, as explained in their respective subsections. However, our definitions are equivalent to the correspondingly named definitions from prior work. Throughout the paper, k denotes the security parameter.

Definition 4.2.1 (Ring signature). *A* ring signature scheme *is a triple of PPT algorithms* $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$*, satisfying the three properties of* correctness *(Definition 4.2.2),* anonymity *(Definitions 4.2.5–4.2.6), and* unforgeability *(Definition 4.2.8). The syntax of* Gen*,* Sign*, and* Verify *follows.*

- $\mathsf{Gen}(1^k)$ *takes k as input and outputs a* verification key vk *and a* signing key sk.
- $\mathsf{Sign}(R, sk, m)$ *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 σ. The set R is also known as a "ring."

- $\mathsf{Verify}(R, \sigma, m)$ *takes as input a set R of verification keys, a signature σ, and a message m, and outputs a single bit indicating whether or not σ 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.Verify *to denote the* Gen, Sign, Verify *algorithms belonging to* RS.

Definition 4.2.2 (Correctness). *A ring signature scheme* RS = (Gen, Sign, Verify) *satisfies* correctness *if there is a negligible function ε such that for any $N = \mathsf{poly}(k)$, any N key pairs $(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k)$, any $i \in [N]$, and any message m,*

$$\Pr\left[\mathsf{Verify}(R, \mathsf{Sign}(R, sk_i, m), m) = 1\right] = 1 - \varepsilon(k) \ , \tag{4.1}$$

where $R = \{vk_1, \ldots, vk_N\}$. RS *satisfies* perfect correctness *if* (4.1) *holds for $\varepsilon = 0$.*

Remark 4. Definition 4.2.2 considers only for honestly generated keys. One could also consider a stronger requirement that verification be successful for honestly generated signatures with respect to rings containing adversarial keys. Any scheme satisfying Definition 4.2.2 can be transformed into one satisfying the stronger definition, by modifying original signature algorithm Sign to a new algorithm Sign$'$ that operates as follows. On input R, sk, m, for a sufficiently large polynomial p:

1. $\sigma \leftarrow \mathsf{Sign}(R, sk, m)$
2. $b_1, \ldots, b_p \leftarrow \mathsf{Verify}(R, \sigma, m)$
3. if $\forall i \in [p], b_i = 1$, output σ; else, go back to step 1

4.2.1 Anonymity

Prior work, notably [RST01, BKM09], has presented a variety of anonymity definitions for ring signatures. Two of the definitions from prior work are relevant to this paper: anonymity against adversarially chosen keys, and anonymity against full key exposure.

This section presents a new, generalized anonymity definition which is parametrized by *oracle sets*, and expresses the two relevant anonymity definitions as instantiations of the generalized definition. This generalized definition is useful to consolidate the existing definitions and make clear their relationship to one another; it captures not only the two definitions we rely on here, but also others from prior work. Moreover, the generalized definition will be essential to concisely express the new anonymity definitions that we introduce in later sections for anonymity of *repudiable* and *claimable* ring signature schemes (in Sections 4.3.1 and 4.3.3 respectively). In a nutshell, this is because the new definitions need to allow the adversary access to additional oracles related to repudiation and/or claiming.

The generalized definition follows. It is parametrized by sets of oracles $\mathcal{O}_1, \mathcal{O}_2$ and an additional parameter $\alpha \in \{0, 1, 2\}$ that limits the adversary's corruptions.

Definition 4.2.3 (($\mathcal{O}_1, \mathcal{O}_2, \alpha$)-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 $\mathsf{Corr}_{(vk_1,sk_1),\dots,(vk_N,sk_N)}$ to take as input $i \in [N]$ and output $\omega_i \leftarrow \mathsf{Gen}^{-1}(vk_i, sk_i)$.*[6]

A ring signature scheme $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ satisfies ($\mathcal{O}_1, \mathcal{O}_2, \alpha$)-anonymity *if for any PPT adversary $\mathcal{A}$ and any polynomial $N = \mathsf{poly}(k)$, $\Pr[b' = b]$ in the above game is negligibly close to $1/2$. That is, formally, $\forall$ PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $N = \mathsf{poly}(k)$, there is a*

[6]The function Gen^{-1} takes as input a verification key vk and signing key sk produced by Gen, and produces the randomness used by Gen to produce this key pair. That is, it samples from the set $\{\omega : \mathsf{Gen}(1^k; \omega) = (vk, sk)\}$. In practice we will only ever invoke Gen^{-1} on a key pair produced by Gen, so we could invert efficiently by simply remembering the randomness used by Gen, but for the purposes of this definition we will describe it as a sampling procedure. Upon the first invocation on an input i, Corr samples $\omega_i \leftarrow \mathsf{Gen}^{-1}(vk_i, sk_i)$, stores it, and outputs it. If Corr is queried twice on the same input i then it outputs the same ω_i that was previously stored.

negligible function ε such that

$$\Pr\left[\begin{array}{ll} (vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k) & \\ ((m^*, i_0^*, i_1^*, R^*), \mathfrak{s}) \leftarrow \mathcal{A}_1^{\mathcal{O}_1, \mathsf{Corr}}(vk_1, \ldots, vk_N) & \\ b \leftarrow \{0,1\} & : \; b' = b \wedge |\{i_0^*, i_1^*\} \cap I| \leq \alpha \\ \sigma \leftarrow \mathsf{Sign}(R^* \cup \{vk_{i_0^*}, vk_{i_1^*}\}, sk_{i_b^*}, m^*) & \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2, \mathsf{Corr}}(\mathfrak{s}, \sigma) & \end{array}\right] < \frac{1}{2} + \varepsilon(k), \tag{4.2}$$

where I is the set of queries to the corruption oracle; and the notation $\mathcal{A}^{\mathcal{O}, \mathsf{Corr}}$ means that for each oracle O in $\mathcal{O}$, $\mathcal{A}$ has oracle access to $O_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$, and $\mathcal{A}$ also has oracle access to $\mathsf{Corr}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$.

Definitions 4.2.5 and 4.2.6 are instantiations of Definition 4.2.3. They are equivalent to the correspondingly named definitions in [BKM09].

Definition 4.2.4 (Signing oracle OSign). *For a ring signature scheme* RS, *the oracle* $\mathsf{OSign}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$ *is defined to take as input $i \in [n]$, a message m, and a set R, and output* $\mathsf{RS.Sign}(R \cup \{vk_i\}, sk_i, m)$. *When the oracle is invoked with respect to a single key pair (i.e.,* $\mathsf{OSign}_{(vk,sk)}$*), we treat the oracle as taking only two inputs, m and R, since i is superfluous in this case.*

Definition 4.2.5 (Anonymity against adversarially chosen keys). *A ring signature scheme* $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies* anonymity against adversarially chosen keys *if it is* $(\{\mathsf{OSign}\}, \varnothing, 0)$*-anonymous. Moreover,* RS *satisfies* adaptive anonymity against adversarially chosen keys *if it is* $(\{\mathsf{OSign}\}, \{\mathsf{OSign}\}, 0)$*-anonymous.*

Definition 4.2.5 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. One can also consider an even stronger definition where the adversary may corrupt *all but one* or even *all* of the parties in the ring, as in Definition 4.2.6.

Definition 4.2.6 (Anonymity against full key exposure). *A ring signature scheme* $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies* anonymity against full key exposure *if it is* $(\{\mathsf{OSign}\}, \varnothing, 2)$*-anonymous.*

Remark 5. Adaptive variants of anonymity were not given or discussed in prior work. [7] In this paper, we refer primarily to *adaptive anonymity against adversarially chosen keys*: this is the strongest notion compatible with repudiability and claimability. To see this, observe that knowledge of a single one (say, $sk_{i_0^*}$) of the two challenge secret keys is sufficient to violate anonymity in a repudiable or claimable scheme, since the challenge signature σ was produced by $sk_{i_0^*}$ if and only if repudiating (resp. claiming) σ using $sk_{i_0^*}$ yields an invalid repudiation (resp. valid claim).

Definition 4.2.6 does not include an adaptive version because adaptivity does not give the adversary any additional power when he can corrupt all the keys.[8]

4.2.2 Unforgeability

The first unforgeability definition that follows is parametrized by an oracle set, taking a similar approach to our anonymity definitions above. In this section, we only give one instantiation of the parametrized definition of unforgeability. We will give other instantiations of Definition 4.2.7 later in the paper, in defining unforgeability for *repudiable* and *claimable* ring signature schemes (in Sections 4.3.1 and 4.3.3 respectively).

Definition 4.2.7 ($\mathcal{O}$-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* $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $\mathcal{O}$-unforgeable *if for any PPT $\mathcal{A}$ and any $N = \mathsf{poly}(k)$, there is a negligible function ε such*

[7]This notwithstanding, the prior constructions of [RST01, BKM09, BK10] all achieve adaptive anonymity. The constructions of [RST01, BK10] achieve perfect/statistical anonymity, respectively — the former, in the random oracle model — so adaptivity follows. The proof of anonymity of [BKM09] construction essentially suffices to prove adaptive anonymity, though the argument was not made in that paper since there was no adaptive definition of anonymity.

[8]An intermediate notion between Definitions 4.2.5 (which sets $\alpha = 0$) and 4.2.6 (which sets $\alpha = 2$) is *anonymity against attribution attacks*, defined in [BKM09], which effectively sets $\alpha = 1$. Adaptive anonymity against attribution attacks is not equivalent to the non-adaptive variant of the same.

that

$$\Pr\left[\begin{array}{l}(vk_1,sk_1),\dots,(vk_N,sk_N)\leftarrow\mathsf{Gen}(1^k)\\(R^*,m^*,\sigma^*)\leftarrow\mathcal{A}^{\mathcal{O},\mathsf{OSign},\mathsf{Corr}}(vk_1,\dots,vk_N)\\b\leftarrow\mathsf{Verify}(R^*,\sigma^*,m^*)\end{array}\quad:\quad\begin{array}{l}b=1\wedge R^*\subseteq\{vk_1,\dots,vk_N\}\setminus I\\\wedge\, Q\cap\{(\cdot,m^*,R^*)\}=\varnothing\end{array}\right]<\varepsilon(k)\ ,$$

where the notation $\mathcal{A}^{\mathcal{O},\mathsf{OSign},\mathsf{Corr}}$ is defined as in Definition 4.2.3, 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 4.2.8 (Unforgeability of ring signatures). *A ring signature scheme* $\mathsf{RS} = (\mathsf{Gen},\mathsf{Sign},\mathsf{Verify})$ *is* unforgeable *if it is $\varnothing$-unforgeable.*[9]

4.3 New definitions for ring signatures: (un)repudiability and (un)claimability

4.3.1 Repudiable ring signatures

Repudiability addresses the question of whether or not members of a ring can prove that they did *not* sign a particular message (when they in fact did not sign it).

Definition 4.3.1 (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 4.2.2),* repudiability *(Definition 4.3.3),* anonymity *(Definition 4.3.4), and* unforgeability *(Definition 4.3.5). The syntax of* Repudiate *and* VerRepud *follows.*

- $\mathsf{Repudiate}(R, sk, \sigma)$ *takes as input a signing key sk, a ring signature σ, and a set of verification keys $R = \{vk_1,\dots,vk_N\}$, and outputs a repudiation ξ.*

[9]This is the definition described as *unforgeability with respect to insider corruption* in [BKM09], and is the strongest of the three unforgeability definitions considered therein.

- $\mathsf{VerRepud}(R, vk, \sigma, \xi)$ *takes as input a set R of verification keys, a signature σ, a repudiation ξ, and an identity vk, and outputs a single bit indicating whether or not ξ is a valid repudiation of signature σ for identity vk.*

Definition 4.3.2 (Repudiation oracle ORpd). *For a repudiable ring signature scheme* RS, *the oracle* $\mathsf{ORpd}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$ *is defined to take as input $i \in [n]$, a signature σ, and a set R, and output* $\mathsf{RS.Repudiate}(R \cup \{vk_i\}, sk_i, \sigma)$. *When the oracle is invoked with respect to a single key pair (i.e.,* $\mathsf{ORpd}_{(vk,sk)}$*), we treat the oracle as taking only two inputs, σ and R, since i is superfluous in this case.*

Additionally, we define the oracle $\mathsf{ORpd}^{\langle\sigma^*\rangle}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$ *to output $\perp$ when it receives the signature σ^* as input, and otherwise to give the same response as* $\mathsf{ORpd}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$.

Repudiability requires two conditions, expressed by equations (4.3) and (4.4) below. Intuitively, (4.3) 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. (4.4) 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.

Definition 4.3.3 (Repudiability). *A ring signature scheme* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies* repudiability *if equipped with algorithms* (Repudiate, VerRepud) *such that the following conditions hold.*

1. (Non-signers can repudiate) *Let $\mathcal{O} = \{\mathsf{OSign}\}$. For any (possibly adversarial) PPT*

signing algorithm $\mathcal{A}_{\mathsf{Sign}}$, there exists a negligible function ε such that

$$\Pr\left[\begin{array}{l} (vk, sk) \leftarrow \mathsf{Gen}(1^k) \\ (\sigma, m, R') \leftarrow \mathcal{A}_{\mathsf{Sign}}^{\mathcal{O},\mathsf{ORpd}_{(vk,sk)}}(vk) \\ \xi \leftarrow \mathsf{Repudiate}(R', sk, \sigma) \\ b \leftarrow \mathsf{VerRepud}(R', vk, \sigma, \xi) \\ b' \leftarrow \mathsf{Verify}(R', \sigma, m) \end{array} : \begin{array}{c} b = 1 \vee b' = 0 \\ \vee Q \cap \{(\cdot, m, R')\} \neq \varnothing \end{array}\right] > 1 - \varepsilon(k) \ . \tag{4.3}$$

2. (Signer cannot repudiate) *For any (possibly adversarial) sign-and-repudiate algorithm $\mathcal{A}_{\mathsf{S\&R}}$, there is a negligible function ε such that for any $N = \mathsf{poly}(k)$,*

$$\Pr\left[\begin{array}{l} (vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k) \\ (\sigma, R', m, \{\xi_{vk}\}_{vk \in R' \setminus R}) \leftarrow \mathcal{A}_{\mathsf{S\&R}}^{\mathcal{O}}(R) \\ \forall vk \in R' \setminus R,\ b_{vk} \leftarrow \mathsf{VerRepud}(R', vk, \sigma, \xi_{vk}) \\ b' \leftarrow \mathsf{Verify}(R', \sigma, m) \end{array} : \begin{array}{c} R' \cap R = \varnothing \vee \\ \bigvee_{vk \in R' \setminus R} b_{vk} = 0 \\ \vee b' = 0 \vee \\ Q \cap \{(\cdot, m, R')\} \neq \varnothing \end{array}\right] > 1 - \varepsilon(k) \ , \tag{4.4}$$

where $R = \{vk_1, \ldots, vk_N\}$, $\mathcal{O} = \{\mathsf{OSign}, \mathsf{ORpd}\}$, and Q is the set of queries to OSign.

Remark 6. Equation 4.4 guarantees that a party possessing a set of signing keys cannot repudiate under all of these keys, *as long as some key in the ring is honestly generated.* If the adversary generates all keys in the ring, then he may be able to produce a repudiation under every key in the ring. However, this does not undermine the purpose of repudiability: indeed, if presented with repudiations under every key in a ring, one can confidently conclude that all keys in the ring were generated dishonestly, and thus that all parties in the ring effectively colluded to produce each signature under that ring. Similarly, given repudiations for a subset of the identities in a ring, one can conclude that *either* one of the remaining identities in the ring produced the signature *or* all of the remaining identities in the ring colluded maliciously to produce the signature. That is, either way, at least one of the remaining identities is responsible for the signature.

Anonymity and unforgeability of repudiable ring signatures

The definitions of anonymity and unforgeability need to be adapted for repudiable ring signature schemes, to incorporate a repudiation oracle as described next.

Definition 4.3.4 (Anonymity of repudiable ring signatures). *A repudiable ring signature scheme*

$$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Repudiate}, \mathsf{VerRepud}))$$

satisfies anonymity against adversarially chosen keys *if* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $(\{\mathsf{OSign}, \mathsf{ORpd}\}, \varnothing, 0)$*-anonymous (Definition 4.2.3) Moreover, the repudiable ring signature satisfies* adaptive *anonymity against adversarially chosen keys if* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $(\{\mathsf{OSign}, \mathsf{ORpd}\}, \{\mathsf{OSign}, \mathsf{ORpd}^{\langle\sigma\rangle}\}, 0)$*-anonymous, where* σ *is the challenge signature in the anonymity experiment (Equation 4.2).*

Recall from Remark 5 that adaptive anonymity against adversarially chosen keys is the strongest anonymity notion compatible with repudiability.

Definition 4.3.5 (Unforgeability of repudiable ring signatures). *A repudiable ring signature scheme*

$$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Repudiate}, \mathsf{VerRepud}))$$

is unforgeable if $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $\{\mathsf{ORpd}\}$*-unforgeable (Definition 4.2.7).*

4.3.2 Unrepudiable ring signatures

We next consider a notion where it is *not* possible for a party to prove to others that he did not produce a particular signature. In fact, though it may not be immediately apparent, a natural formalization of this notion is expressed by the definition of *anonymity against full key exposure* (Definition 4.2.6): that is, the strongest of the anonymity definitions given in Section 4.2. The following paragraphs justify this claim with detailed intuition.

Recall that anonymity against full key exposure (FKE) preserves signer anonymity even against an adversary that obtains all of the secret keys of all members of a ring. A ring signature scheme that satisfies repudiability could not also satisfy anonymity against FKE, because of the following attack: the adversary obtains all secret keys in the ring, attempts to repudiate using each secret key, and identifies as the signer the one secret key with respect to which the repudiation algorithm does not produce a valid repudiation. With overwhelming probability, by definition of repudiability, there is exactly one such secret key.

This informal argument establishes that anonymity against FKE must imply any reasonable notion of unrepudiability. The question then arises: are the two notions equivalent? While there arguably exist meaningful definitions of unrepudiability that are weaker than anonymity against FKE, we believe anonymity against FKE is the most reasonable definition of unrepudiability, as explained next.

Any reasonable definition of unrepudiability should capture the intuitive requirement that non-signers cannot behave distinguishably from signers. A little more precisely, for any protocol that could be executed by a non-signer Nancy with respect to a signature σ and her verification key vk', the signer Sigmund of that signature must be able to engage in the same protocol with respect to his own verification key vk and behave indistinguishably from Nancy. In other words, we require that if Nancy's secret key were stolen, the thief would be unable to tell whether σ was produced by Nancy or by someone else. Indeed, if Nancy were stateless and did not remember what signatures she had produced in the past, or simply lent her secret key to someone else who used it to produce signatures, then she herself would not be able to tell. The definition of anonymity against FKE embodies almost exactly this requirement — but instead of requiring anonymity against the thief who steals just Nancy's key, the definition makes the stronger requirement that anonymity must hold even against a thief who has every secret key in the ring corresponding to σ.

Is a weaker definition, which only rules out *unilateral* repudiations by a single party, a meaningful definition of unrepudiability? Perhaps. However, it is more in keeping with the intuitive goals and standard properties of ring signatures to protect against adversaries that may have many or all secret keys in a ring: that is, to rule out even the possibility of multiple ring members *colluding* to produce a repudiation for some ring member. Thus we arrive at the following definition.

Definition 4.3.6 (Unrepudiable ring signature scheme). *A ring signature scheme satisfies* unrepudiability *if it satisfies anonymity against full key exposure (Definition 4.2.6).*

4.3.3 Claimable ring signatures

Claimability addresses whether the actual signer can prove later that they were the signer, without remembering the signing randomness.

Definition 4.3.7 (Claimable ring signature). *A* claimable ring signature scheme *is a ring signature scheme with an additional pair of algorithms* $(\mathsf{Claim}, \mathsf{VerClaim})$*, satisfying the four properties of* correctness *(Definition 4.2.2),* claimability *(Definition 4.3.9),* anonymity *(Definition 4.3.10), and* unforgeability *(Definition 4.3.11). The syntax of* Claim *and* $\mathsf{VerClaim}$ *follows.*

- $\mathsf{Claim}(R, sk, \sigma)$ *takes as input a signing key sk, a ring signature* σ*, and a set of verification keys* $R = \{vk_1, \ldots, vk_N\}$*, and outputs a claim* ζ*.*
- $\mathsf{VerClaim}(R, vk, \sigma, \zeta)$ *takes as input a set* R *of verification keys, a signature* σ*, a claim* ζ*, and an identity vk, and outputs a single bit indicating whether or not* ζ *is a valid claim of signature* σ *for identity vk.*

Definition 4.3.8 (Claim oracle OClaim). *For a claimable ring signature scheme* RS*, the oracle* $\mathsf{OClaim}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$ *is defined to take as input* $i \in [n]$*, a set* R*, and a signature* σ*, and output* $\mathsf{RS.Claim}(R, sk, \sigma)$*. When the oracle is invoked with respect to a single key pair (i.e.,* $\mathsf{OClaim}_{(vk,sk)}$*), we treat the oracle as taking only two inputs,* R *and* σ*, since* i *is superfluous in this case.*

Additionally, we define the oracle $\mathsf{OClaim}^{\langle\sigma^*\rangle}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$ *to output* $\perp$ *when it receives the signature* σ^* *as input, and otherwise to give the same response as* $\mathsf{OClaim}_{(vk_1,sk_1),\ldots,(vk_N,sk_N)}$.

Claimability requires three conditions, expressed by equations (4.5), (4.6), and (4.7) below. Informally, (4.5) requires that honest signers can successfully claim their signatures, (4.6) requires that adversarial parties cannot successfully claim a signature that they did not produce, and (4.7) requires that adversarial parties cannot produce a signature along with a claim that appears to be produced by an honest party (that is, falsely framing the honest party as the signer).

Definition 4.3.9 (Claimability). *A ring signature scheme* (Gen, Sign, 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 ε such that for any $N = \mathsf{poly}(k)$ and $(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k)$ and any $i \in [N]$, it holds for any message m that*

$$\Pr\left[\sigma \leftarrow \mathsf{Sign}(R, sk_i, m) : \mathsf{VerClaim}(R, vk_i, \sigma, \mathsf{Claim}(R, sk_i, \sigma)) = 1\right] > 1 - \varepsilon(k), \quad (4.5)$$

where $R = \{vk_1, \ldots, vk_N\}$.[10]

2. (Non-signers cannot claim) *Let $\mathcal{O} = \{\mathsf{OSign}\}$. For any (possibly adversarial) PPT sampling-and-claiming algorithm $\mathcal{A}_{\mathsf{Claim}} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ε such that*

$$\Pr\left[\begin{array}{lcl} (vk, sk) \leftarrow \mathsf{Gen}(1^k) & & \\ (R', m, \mathfrak{s}) \leftarrow \mathcal{A}_1^{\mathcal{O}, \mathsf{OClaim}_{(vk,sk)}}(vk) & & \\ \sigma \leftarrow \mathsf{Sign}(R' \cup \{vk\}, sk, m) & & b = 1 \wedge b' = 1 \\ (\zeta, vk') \leftarrow \mathcal{A}_2^{\mathcal{O}, \mathsf{OClaim}_{(vk,sk)}}(R' \cup \{vk\}, \sigma, \mathfrak{s}) & : & \wedge vk' \neq vk \\ b \leftarrow \mathsf{VerClaim}(R' \cup \{vk\}, vk', \sigma, \zeta) & & \\ b' \leftarrow \mathsf{Verify}(R' \cup \{vk\}, \sigma, m) & & \end{array}\right] < \varepsilon(k). \quad (4.6)$$

3. (Malicious signer cannot frame an honest party) *For any PPT adversary $\mathcal{A}_{\mathsf{S\&C}}$, there*

[10]Like Definition 4.2.2, Equation 4.5 considers only honestly generated keys. See Remark 4 for further discussion.

exists a negligible function ε such that

$$\Pr\left[\begin{array}{lcl} (vk, sk) \leftarrow \mathsf{Gen}(1^k) & & \\ (R', m, \sigma, \zeta) \leftarrow \mathcal{A}_{\mathsf{S\&C}}^{\mathcal{O},\mathsf{OClaim}_{(vk,sk)}}(vk) & & b = 1 \wedge b' = 1 \\ b \leftarrow \mathsf{VerClaim}(R' \cup \{vk\}, vk, \sigma, \zeta) & : & \wedge Q \cap \{(\cdot, \sigma)\} = \varnothing \\ b' \leftarrow \mathsf{Verify}(R' \cup \{vk\}, \sigma, m) & & \end{array}\right] < \varepsilon(k). \quad (4.7)$$

where $\mathcal{O} = \{\mathsf{OSign}\}$ and Q is the set of queries made to oracle $\mathsf{OClaim}_{vk,sk}$.

Remark 7. Our definition and construction guarantee that all honestly generated signatures can be claimed by the signer, and even malicious signers cannot frame non-signers. Our definition does not guarantee that all signatures that verify can be claimed by someone; requiring this would be a reasonable alternative definition, but we believe our claimability definition is reasonable and sufficient in many settings. Since claimability is a notion designed for the signer's benefit, it is not an "attack" that the signer can choose to waive the ability to claim. If two ring members collude to produce a signature, then they are effectively both signers; therefore, it is not an attack if two signers collude to produce a signature claimable by one of them.

Our definition has a property that may be counterintuitive at first glance: namely, a single party might be able to claim and repudiate the same signature. However, this possibility is in fact fully consistent with our security guarantees, as follows. If the party does both actions, we (the observers) know she is adversarial. If she claims, we know that she either signed or was part of an adversarial group that signed. If she repudiates, we know that either she didn't sign, or she was part of an adversarial group of which another member must be unable to repudiate.

Anonymity and unforgeability of claimable ring signatures

The definitions of anonymity and unforgeability must be adapted for claimable ring signature schemes, to allow the adversary a claim oracle as described next.

Definition 4.3.10 (Anonymity of claimable ring signatures). *A claimable ring signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Claim}, \mathsf{VerClaim}))$ *satisfies anonymity against adversarially chosen keys if* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $(\{\mathsf{OSign}, \mathsf{OClaim}\}, \varnothing, 0)$*-anonymous (Definition 4.2.3). Moreover, the repudiable ring signature satisfies* adaptive *anonymity against adversarially chosen keys if* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is*

$$(\{\mathsf{OSign}, \mathsf{OClaim}\}, \{\mathsf{OSign}, \mathsf{OClaim}^{\langle\sigma\rangle}\}, 0)\textit{-anonymous}\ ,$$

where σ *is the challenge signature in the anonymity experiment (Equation* (4.2)*).*

Recall from Remark 5 that adaptive anonymity against adversarially chosen keys is the strongest anonymity notion compatible with claimability.

Definition 4.3.11 (Unforgeability of claimable ring signatures). *A claimable ring signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Claim}, \mathsf{VerClaim}))$ *is* unforgeable *if* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $\{\mathsf{OClaim}\}$*-unforgeable (Definition 4.2.7).*

4.3.4 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. The result is that even an adversary holding all ring members under duress cannot figure out who produced a given signature. This is true even if the ring members under duress attempt to cooperate with the adversary.

To achieve this, 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 4.3.12 (Unclaimable ring signatures). *A* unclaimable *ring signature scheme is a ring signature scheme augmented with an additional algorithm* $\mathsf{ExtractRandomness}$ *as follows.*

- $\mathsf{ExtractRandomness}(R, sk, \sigma, m)$ *takes as input a ring R, a secret key sk, a signature σ and a message m. If sk is one of the secret keys for ring R, and σ is a signature on m with respect to R, then it outputs randomness ρ.*

$\mathsf{ExtractRandomness}$ *must satisfy the following condition.*

- (Statistical unclaimability) *Let $\mathcal{R}$ be the distribution of signing randomness. For any $N = \mathsf{poly}(k)$ there is a negligible function ε such that the following holds. Let $(vk_1, sk_1), (vk_2, sk_2) \leftarrow \mathsf{Gen}(1^k)$. For any message m and any $vk_3, \ldots, vk_N$ and $sk_3, \ldots, sk_N$, let $R = \{vk_1, \ldots, vk_N\}$ and $S = \{(i, vk_i, sk_i)\}_{i \in [N]}$. Let $\rho \leftarrow \mathcal{R}$, $\sigma_1 \leftarrow \mathsf{Sign}(R, sk_1, m; \rho)$, and $\rho_1 \leftarrow \mathsf{ExtractRandomness}(R, sk_2, \sigma_1, m)$. Let $\rho_2 \leftarrow \mathcal{R}$ and $\sigma_2 \leftarrow \mathsf{Sign}(R, sk_2, m; \rho_2)$. Then*

$$(S, \rho_1, \sigma_1) \approx_\varepsilon (S, \rho_2, \sigma_2).$$

Definition 4.3.12 is unusual among the definitions in this paper, in that it gives a statistical rather than a computational guarantee. We opted to give the statistical definition because it is simpler, it is a stronger guarantee, and our construction in this case achieves the statistical guarantee. One could also consider a computational definition.

Remark 8 (Claimability is not the opposite of unclaimability). According to these definitions, unclaimability is *not* technically the opposite of claimability (even when ignoring the fact that the formal definitions give a statistical guarantee for unclaimability but a computational guarantee for claimability). Claimability requires the ability to "voluntarily claim" a signature *without remembering the signing randomness*, whereas unclaimability rules out the ability to "claim under duress" *even given the signing randomness.* For voluntary claims, the natural and stronger definition is to guarantee the ability to claim adaptive, without "planning ahead" and without the storage requirement of remembering the signing randomness. In contrast, when considering attempts to claim under duress, the natural and stronger definition is to rule out the possibility of successful claims even in the presence of the signing randomness.

Remark 9 (Unclaimability protects *honest* signers). An adversarial signer who *wants* to claim can devise ways of credibly later claiming a ring signature, even when using an unclaimable ring signature scheme.[11] This does not decrease the utility of an unclaimable ring signaature scheme for honest signers who *want* their signatures to be unclaimable.

Remark 10 (Secure erasure and rubber-hose adversaries). Our unclaimability definition does not protect signers against adversaries that can unexpectedly compromise the internal state of ring members, in the following sense: either the signer will have securely erased the signing randomness, in which case the adversary will not be able to determine the signer's identity; or she won't have erased the signing randomness, in which case the adversary will be able to distinguish the signer from other ring members. In the first case, the signer's anonymity comes from her diligent erasure, rather than from unclaimability; and the first case is only possible assuming secure erasure is feasible.

We believe our unclaimability definition still has twofold value. First, even assuming secure erasure, a system is stronger if it eliminates reliance on users to erase diligently. Users often fail to use systems as they are supposed to; this is a major source of real-world security failures. Our definition's security does not rely on any user actions beyond running the honest signing algorithm.

Secondly, unclaimability is useful beyond the context of such powerful, real-time adversaries. Consider an honest signer who does not intend to claim at the time of signing, but later is corrupted or changes her mind. Our definition guarantees that the signer's intent at the time of signing is what matters: if she intended unclaimability when signing, she cannot later claim that signature.

Unclaimability implies unrepudiability

Any unclaimable ring signature scheme is also unrepudiable. Recall that the definition of unclaimability captures the idea that for any function that the true signer can compute given

[11]For example, an adversarial signer might use a PRG output as his signing randomness, or append it to his message, and remember the preimage. If he later revealed the preimage, it would likely serve as a credible claim to authorship of the signature.

the signing randomness and the secret key, any other member of the ring can compute an indistinguishable function. Intuitively, the implication follows from the fact that repudiation would require a non-signer to behave in a way that *distinguishable* from any possible behavior of the actual signer.

Theorem 4.3.1. *Any unclaimable ring signature scheme is also unrepudiable.*

Proof. Recall that unrepudiability (Definition 4.3.6) is defined as anonymity against full key exposure (Definition 4.2.6), which is $(\{\mathsf{OSign}\}, \varnothing, 2)$-anonymity under the framework of Definition 4.2.3. Thus, a ring signature scheme is unrepudiable if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and polynomial N, it satisfies (4.2) of Definition 4.2.3 for $(\mathcal{O}_1, \mathcal{O}_2, \alpha) = (\{\mathsf{OSign}\}, \varnothing, 2)$. Note that since $\alpha = 2$, we can consider without loss of generality only adversaries that use the corruption oracle to learn all N secret keys in (4.2), and do not make any queries to the signing oracle (since the adversary can produce signatures himself, using the secret keys).

To establish the theorem, it suffices to show that for any unclaimable ring signature scheme, the view of the adversary $\mathcal{A}$ is indistinguishable between the cases $b = 0$ and $b = 1$. Unclaimability (Definition 4.3.12) directly implies that these two views are indistinguishable. To see this, recall that the adversary's inputs in (4.2) of Definition 4.2.3 are an honestly generated set of verification keys $vk_1, \ldots, vk_N$ and a signature σ produced by the honest signing algorithm using a secret key corresponding to some $vk_i, i \in [N]$. The theorem follows. □

Remark 11. It is unclear that the reverse implication holds even for a computational definition of unclaimability. The main complication is that unclaimability must hold even if the signing randomness is saved, while this is not an issue for unrepudiability. For instance, an algorithm that appends a commitment to the signing randomness (or to a random nonce) could be unrepudiable, but could be claimed by a signer who remembered the signing randomness.

4.3.5 Repudiable-and-claimable ring signatures

Suppose that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is a ring signature scheme, and there are algorithms $\mathsf{Repudiate}$, $\mathsf{VerRepud}$, Claim, and $\mathsf{VerClaim}$ such that

$$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Repudiate}, \mathsf{VerRepud})) \quad \text{and} \quad (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Claim}, \mathsf{VerClaim}))$$

are a repudiable ring signature scheme and a claimable ring signature scheme respectively (Definitions 4.3.1 and 4.3.7). 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 4.3.1 (repudiability) does *not* allow the adversary a claim oracle, and likewise Definition 4.3.7 (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 4.3.1 and 4.3.7 is defined with respect to a *quintuple* of algorithms either containing $\mathsf{Repudiate}$ but not Claim, or vice versa, the concept of the "other oracle" is undefined within the scope of each definition.

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 1. *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 η_0, η_1 such that η_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 η_0, η_1.* $\mathsf{Repudiate}$ *produces repudiations just as in the original scheme, but additionally appends η_0 to every repudiation.* Claim *produces claims just as in the original scheme, but additionally appends η_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. As the resulting formal definitions are somewhat repetitive, we defer them to Appendix 4.7.

4.4 Repudiable construction

4.4.1 Building blocks

ZAPs ZAPs are two-message public coin witness indistinguishable proofs [DN07b].

Definition 4.4.1 (ZAP). *A ZAP for an NP language L with witness relation $\mathcal{R}_L$ is a triple of algorithms $\mathsf{ZAP}_L = (\mathsf{ZAP.Setup}_L, \mathsf{ZAP.Prove}_L, \mathsf{ZAP.Verify}_L)$, where $\mathsf{ZAP.Setup}$ and $\mathsf{ZAP.Prove}$ are PPT and $\mathsf{ZAP.Verify}$ is polynomial-time and deterministic, satisfying the following properties.*

Public coin. *For some polynomial $\ell = \ell(k)$, $\mathsf{ZAP.Setup}$ is the algorithm that on input 1^k, outputs a uniformly random element of $\{0,1\}^\ell$.*

Completeness. *For $(x, w) \in \mathcal{R}_L$ and any $\rho \in \{0,1\}^{\ell(k)}$ we have*

$$\Pr_{\pi \leftarrow \mathsf{ZAP.Prove}(\rho,x,w)}[\mathsf{ZAP.Verify}(\rho, \pi, x) = 1] = 1.$$

Adaptive soundness. *There exists a negligible function ε such that*

$$\Pr_{\rho \leftarrow \mathsf{ZAP.Setup}(1^k)}[\exists(x, \pi) : x \notin L \wedge \mathsf{ZAP.Verify}(\rho, \pi, x)] \leq \varepsilon(k).$$

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 $(x_k, w_{0,k}), (x_k, w_{1,k}) \in \mathcal{R}_L$,* *the following pair of ensembles is computationally indistinguishable:*

$$\{\mathsf{ZAP.Prove}(\rho_k, x_k, w_{0,k})\}_{k\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{ZAP.Prove}(\rho_k, x_k, w_{1,k})\}_{k\in\mathbb{N}} .$$

In this work, for simplicity, we will assume use of a ZAP for some NP-complete language L_{NP} (with witness relation $\mathcal{R}_{L_{\mathrm{NP}}}$) and for any $L \in \mathrm{NP}$ with witness relation $\mathcal{R}_L$, we define $\mathsf{ZAP.Prove}_L$ and $\mathsf{ZAP.Verify}_L$ as follows.

- $\mathsf{ZAP.Prove}_L$ takes as input a triple (ρ, x, w). If $(x, w) \notin \mathcal{R}_L$, then output $\perp$. Otherwise, use an NP reduction on (x, w) to get a pair $(x_{\mathrm{NP}}, w_{\mathrm{NP}}) \in \mathcal{R}_{L_{\mathrm{NP}}}$, and output $\mathsf{ZAP.Prove}(\rho, x, w)$.

- $\mathsf{ZAP.Verify}_L$ takes as input a triple (ρ, π, x), uses the same NP reduction to obtain x_{NP} (which is in L_{NP} iff $x \in L$), and outputs $\mathsf{ZAP.Verify}(\rho, \pi, x)$.

Verifiable random functions (VRFs) [MRV99] are another main building block of our construction. 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 4.4.2 (VRF). *A* verifiable random function *(VRF) is a tuple of algorithms* $\mathsf{VRF} = (\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Prove}, \mathsf{VRF.Verify})$*, where* Gen *and* Verify *are PPT and* Eval *and* Prove *are polynomial time and deterministic, satisfying:*

Complete provability *With probability at least* $1 - 2^{-\Omega(k)}$ *over* $(pk, sk) \leftarrow \mathsf{VRF.Gen}(1^k)$*, we have for all inputs* x *that*

$$\Pr[\mathsf{VRF.Verify}(pk, x, \mathsf{VRF.Eval}(sk, x), \mathsf{VRF.Prove}(sk, x)) = 1] > 1 - 2^{-\Omega(k)} .$$

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*

$$\Pr[\mathsf{VRF.Verify}(pk, x, y_i, \tau_i) = 1] < 2^{-\Omega(k)}.$$

Residual pseudorandomness *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a probabilistic polynomial-time adversary, where both $\mathcal{A}_1$ and $\mathcal{A}_2$ get oracle access to the VRF evaluation and prove algorithms. Let $(pk, sk) \leftarrow \mathsf{VRF.Gen}(1^k)$, and let*

$$(x, \mathfrak{s}) \leftarrow \mathcal{A}_1^{\mathsf{VRF.Eval}(sk,\cdot),\mathsf{VRF.Prove}(sk,\cdot)}(1^k, pk).$$

Let $b \leftarrow \{0,1\}$, and let v be either $\mathsf{VRF.Eval}(sk, x)$ or uniformly random, depending on the choice bit b. Let

$$b' = \mathcal{A}_2^{\mathsf{VRF.Eval}(sk,\cdot),\mathsf{VRF.Prove}(sk,\cdot)}(1^k, v, \mathfrak{s}).$$

Then there is a negligible function ε such that $\Pr[b = b' \text{ and } x \notin Q] < 1/2 + \varepsilon(k)$, where Q is the set of oracle queries made by $\mathcal{A}$ to either oracle.

For simplicity, we assume that Eval takes inputs x of any length, i.e., $x \in \{0,1\}^*$.

Definition 4.4.3. *The* verification failure probability *of a VRF* VRF *is*

$$\Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathsf{VRF.Gen}(1^k)\\ b \leftarrow \mathsf{VRF.Verify}(pk, x, \mathsf{VRF.Eval}(sk, x), \mathsf{VRF.Prove}(sk, x))\end{array} \;:\; b = 0\right].$$

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 holds for the uncorrupted keys only).

Lemma 4.4.1 (Parallel VRF Game). *Let* VRF *be a a VRF. Then* $\forall$ *PPT* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and* *all $N = \mathsf{poly}(k)$, there is a negligible function ε such that*

$$\Pr\left[\begin{array}{l} (pk_1, sk_1), \ldots, (pk_N, sk_N) \leftarrow \mathsf{VRF.Gen}(1^k) \\ (m^*, \mathfrak{s}) \leftarrow \mathcal{A}_1^{\mathcal{V},\mathsf{Corr}}(vk_1, \ldots, vk_N) \\ \forall i \in [N], y_{i,0} \leftarrow \mathsf{VRF.Eval}(sk_i, m^*) \\ \forall i \in [N], y_{i,1} \leftarrow \$ \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}_2(\mathfrak{s}, (y_{i,b})_{i \in [N] \setminus C}) \end{array} : \begin{array}{l} b = b' \wedge \\ \forall i \in [N] \setminus C, \\ (i, m^*) \notin Q \end{array}\right] < 1/2 + \varepsilon(k) \ , \quad (4.8)$$

where $\mathcal{V}$ is an oracle that takes as input a pair (i, m) and outputs

$$(y, \tau) = (\mathsf{VRF.Eval}(sk_i, m), \mathsf{VRF.Prove}(sk_i, m)) \ ,$$

and 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 queries to $\mathcal{V}$.*

We will refer to the game described by (4.8) as the Parallel VRF Game.

4.4.2 Construction

Construction 4.4.2. *Our construction* R-RS *is parametrized by* ZAP*,* VRF*, and M, where*

- ZAP *is a ZAP;*
- VRF *is a VRF with input domain $\{0,1\}^*$, whose* Verify *algorithm takes ν bits of randomness and whose verification failure probability (Definition 4.4.3) is ε;*[12]
- *M is a polynomial satisfying $M \geq (\nu + k)/\log_2(1/\varepsilon)$.*[13]

We first present the Gen *algorithm of our ring signature scheme* R-RS.

[12]ε need not be explicitly known, as discussed in footnote 13.

[13]This inequality is required in order to invoke Corollary 4.4.6. As explained in Remark 13, a satisfactory value of M can be set even without knowledge of ε. If ε happens to be known, a smaller value of M can be chosen.

R-RS.Gen(1^k)

1. $(vk^1_{\mathsf{VRF}}, sk^1_{\mathsf{VRF}}), \ldots, (vk^4_{\mathsf{VRF}}, sk^4_{\mathsf{VRF}}) \leftarrow \mathsf{VRF.Gen}(1^k)$.
   Let $\vec{vk}_{\mathsf{VRF}} = (vk^1_{\mathsf{VRF}}, \ldots, vk^4_{\mathsf{VRF}})$ *and* $\vec{sk}_{\mathsf{VRF}} = (sk^1_{\mathsf{VRF}}, \ldots, sk^4_{\mathsf{VRF}})$.
2. $\rho \leftarrow \mathsf{ZAP.Setup}(1^k)$.
3. $\vec{\alpha} = (\alpha_1, \ldots, \alpha_M) \leftarrow (\{0,1\}^\nu)^M$.
4. *Output* $vk = (\vec{vk}_{\mathsf{VRF}}, \rho, \vec{\alpha})$ *and* $sk = (\vec{sk}_{\mathsf{VRF}}, vk)$.[14]

In the rest of the section, we (implicitly) use the following convention to parse a ring R.

$$\begin{aligned}&\textit{Write } R = \{vk_1, \ldots, vk_N\}.\\ &\textit{For each } i \in [N], \textit{ write } vk_i = (\vec{vk}^i_{\mathsf{VRF}} = (vk^{i,1}_{\mathsf{VRF}}, \ldots, vk^{i,4}_{\mathsf{VRF}}), \rho_i, \vec{\alpha}_i = (\alpha^i_1, \ldots, \alpha^i_M)).\end{aligned} \tag{4.9}$$

Definition 4.4.4. *Let L be the following NP language.*

$$\begin{aligned}\{(R, m, \varphi, (y_1, y_2, y_3, y_4)) \;:\; \exists i^*, \tau_1, \tau_2, \tau_3, \tau_4, \gamma \textit{ s.t. } (b_1 \vee b_2) \wedge (b_3 \vee b_4)\\ \textit{where } \forall \eta \in \{1,2,3,4\}, b_\eta = \bigwedge_{i\in[N], j\in[M]} \mathsf{VRF.Verify}(vk^{i^*,\eta}_{\mathsf{VRF}}, (R, m, \varphi), y_\eta, \tau_\eta; \alpha^i_j \oplus \gamma)\}\;.\end{aligned}$$

We now present the Sign *and* Verify *algorithms of our construction.*

R-RS.Sign(R, sk, m)

1. *Parse R as described above and* $sk = ((sk^1_{\mathsf{VRF}}, \ldots, sk^4_{\mathsf{VRF}}), vk)$.
2. *If* $vk \notin R$ *output* $\perp$ *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. $\varphi \leftarrow \{0,1\}^k$. *(This is used as a salt for the VRF input in Step 7, and output in Step 8.)*

[14]We include the verification key in sk so that the Sign procedure can identify the verification key in the ring corresponding to the signing key.

6. *For* $\eta \in \{1,2,3,4\}$, *let* $y_\eta = \mathsf{VRF.Eval}(sk^\eta_{\mathsf{VRF}}, (R, m, \varphi))$ *and* $\tau_\eta = \mathsf{VRF.Prove}(sk^\eta_{\mathsf{VRF}}, (R, m, \varphi))$.
   Let $\vec{y} = (y_1, \dots, y_4)$.
7. *For each* $i \in [N]$, *let* $\pi_i \leftarrow \mathsf{ZAP.Prove}_L(\rho_i, (R, m, \varphi, \vec{y}), (i^*, \tau_1, \perp, \tau_3, \perp, \gamma))$.
   Let $\vec{\pi} = (\pi_1, \dots, \pi_N)$.
8. *Output* $\sigma = (\vec{\pi}, \vec{y}, \varphi)$.

$\mathsf{R\text{-}RS.Verify}(R, \sigma, m)$

1. *Parse* R *as above and* $\sigma = ((\pi_1, \cdots, \pi_N), \vec{y}, \varphi)$.
2. *Output* $\bigwedge_{i\in[N]} \mathsf{ZAP.Verify}_L(\rho_i, \pi_i, (R, m, \varphi, \vec{y}))$.

Now that we have described the main algorithms of R-RS, *we proceed to describe the repudiation algorithms for* R-RS.

Definition 4.4.5. *Let* L' *be the following NP language:*

$$\Big\{ (R, m, \varphi, (y_1, \dots, y_4), vk = (\vec{vk}_{\mathsf{VRF}}, \rho, \vec{\alpha})) \ : \ \exists i^*, y'_1, \dots, y'_4, \tau'_1, \dots, \tau'_4, \gamma \text{ s.t.}$$

$$((b'_1 \wedge b'_2) \vee (b'_3 \wedge b'_4)) \wedge vk = vk_{i^*}, \text{ where } \forall \eta \in \{1,2,3,4\},$$

$$b'_\eta = \left(y'_\eta \neq y_\eta \wedge \bigwedge_{i\in[N], j\in[M]} \mathsf{VRF.Verify}(vk^{i^*,\eta}_{\mathsf{VRF}}, (R, m, \varphi), y'_\eta, \tau'_\eta; \alpha^i_j \oplus \gamma) \right) \Big\} .$$

$\mathsf{R\text{-}RS.Repudiate}(R, sk, \sigma)$

1. *Parse* R *as above,* $sk = ((sk^1_{\mathsf{VRF}}, \dots, sk^4_{\mathsf{VRF}}), vk)$, *and* $\sigma = (\vec{\pi}, \vec{y}, \varphi)$.
2. *If* $vk \notin R$ *output* $\perp$ *and halt.*
3. *Define* $i^* \in [N]$ *such that* $vk_{i^*} = vk$.
4. *For* $\eta \in \{1,2\}$: *let* $y'_\eta = \mathsf{VRF.Eval}(sk^\eta_{\mathsf{VRF}}, (R, m, \varphi))$ *and let* $\tau'_\eta = \mathsf{VRF.Prove}(sk^\eta_{\mathsf{VRF}}, (R, m, \varphi))$.
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 $\xi_i \leftarrow \mathsf{ZAP.Prove}_{L'}(\rho_i, (R, m, \varphi, \vec{y}, vk), (i^*, y'_1, y'_2, \bot, \bot, \tau'_1, \tau'_2, \bot, \bot, \gamma))$.

7. *Output* $\xi = (\xi_1, \ldots, \xi_N)$.

$\mathsf{R\text{-}RS.VerRepud}(R, vk, \sigma, \xi)$

1. *Parse* R *as above. If* $vk \notin R$*, output* 1 *and halt.*
2. *Parse* $\sigma = (\vec{\pi}, \vec{y}, \varphi)$*, and* $\xi = (\xi_1, \ldots, \xi_N)$.
3. *Output* $\bigwedge_{i \in [N]} \mathsf{ZAP.Verify}_{L'}(\rho_i, \xi_i, (R, m, \varphi, \vec{y}, vk))$.

Remark 12. As written, the size of the VRF input (R, m, φ) scales with the size of the ring R, and we have assumed that the VRF has input domain $\{0,1\}^*$, 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, φ) rather than on (R, m, φ) directly. We have presented the version of the scheme without the hash function, for simplicity of exposition.

4.4.3 Security proof

Theorem 4.4.3. *Let* VRF *be a VRF and* ZAP *be a ZAP. Then* R-RS *is a repudiable ring signature scheme.*

Proof. Follows directly from Lemmata 4.4.4 (correctness), 4.4.7 (repudiability), 4.4.8 (unforgeability), and 4.4.9 (anonymity). □

Lemma 4.4.4 (Correctness of R-RS). R-RS *satisfies correctness (Definition 4.2.2).*

Correctness is immediate so we omit the proof. Before presenting the proof of repudiability, we establish the following supporting lemma and corollary, which proceed according to an argument of [DN07b].

Lemma 4.4.5. *Let $\mathcal{V}$ be a randomized algorithm that takes ν 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 ε,*

$$\Pr\left[\mathcal{V}(1^k, x) = \beta\right] \geq 1 - \varepsilon \ . \tag{4.10}$$

Let M be a polynomial such that $M \geq (\nu + k)/\log_2(1/\varepsilon)$. (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:

$$\Pr_{(\alpha_1,\ldots,\alpha_M)\leftarrow(\{0,1\}^\nu)^M}\left[\forall\gamma \in \{0,1\}^\nu,\ \exists i \in [M] \ s.t.\ \mathcal{V}(1^k, x; \alpha_i \oplus \gamma) = \beta\right] \ . \tag{4.11}$$

Proof. Fix any $\gamma \in \{0,1\}^\nu$. Let $\psi_{i,\gamma} = \alpha_i \oplus \gamma$ for each $i \in [M]$. Since the α_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 γ. Therefore, conditioned on any given γ,

$$\Pr\left[\forall i \in [M],\ \mathcal{V}(1^k, x, ; \psi_{i,\gamma}) \neq \beta\right] < \varepsilon^M \ .$$

There are 2^ν possible values of γ, so by a union bound,

$$\Pr\left[\exists\gamma \in \{0,1\}^\nu \text{ s.t. } \forall i \in [M],\ \mathcal{V}(1^k, x; \psi_{i,\gamma}) \neq \beta\right] < 2^\nu \cdot \varepsilon^M \ .$$

Since $M \geq (\nu + k)/\log_2(1/\varepsilon) = \log_\varepsilon(2^{-(\nu+k)})$ by assumption, the right-hand side is at most 2^{-k}, which is negligible. The lemma follows. $\square$

The next corollary states that the implication established in Lemma 4.4.5 in fact goes both ways.

Corollary 4.4.6. *Let $\mathcal{V}$ be a randomized algorithm takes ν bits of randomness and outputs one bit. Let $\beta \in \{0,1\}$ be a bit, and let ε be a negligible function. Let M be a polynomial such that $M \geq (\nu + k)/\log_2(1/\varepsilon)$. Then* (4.10) *holds if and only if* (4.11) *is overwhelming.*

Proof. Follows from applying Lemma 4.4.5 for both $\beta = 0$ and $\beta = 1$. $\square$

Remark 13. For all large enough k, Corollary 4.4.6 holds for any $M \geq \nu + k$. This is because if (4.10) holds for some negligible ε, then for all large enough k, (4.10) also holds for $\varepsilon = 1/2$ (or indeed, any constant ε). Substituting $\varepsilon = 1/2$ into $M \geq (\nu + k)/\log_2(1/\varepsilon)$ yields $M \geq \nu + k$. In particular, this means that a satisfactory M can be chosen without knowledge of ε.

Next, we give the proof of repudiability of R-RS.

Lemma 4.4.7 (Repudiability of R-RS). R-RS *is repudiable (Definition 4.3.3).*

Proof. Suppose, for contradiction, that R-RS is not repudiable. Then by Definition 4.3.3, it must be that either (4.3) or (4.4) does not hold. We consider these two possibilities in turn.

Suppose first that (4.3) does not hold for R-RS. Then there is a PPT $\mathcal{A}_{\mathsf{Sign}}$ that generates a valid signature σ with respect to some ring R, so that σ is not repudiable by some honest party in the ring. That is, the following probability is non-negligible:

$$\Pr\left[\begin{array}{l} (vk, sk) \leftarrow \mathsf{Gen}(1^k) \\ (\sigma, m, R) \leftarrow \mathcal{A}^{\mathcal{O}}_{\mathsf{Sign}}(vk) \\ \xi \leftarrow \mathsf{Repudiate}(R, sk, \sigma) \\ b \leftarrow \mathsf{VerRepud}(R, vk, \sigma, \xi) \\ b' \leftarrow \mathsf{Verify}(R, \sigma, m) \end{array} : \begin{array}{c} b = 0 \wedge b' = 1 \\ \wedge Q \cap \{(\cdot, m, R')\} = \varnothing \end{array}\right], \tag{4.12}$$

where Q is the set of queries to OSign.

Based on $\mathcal{A}_{\mathsf{Sign}}$, we build an adversary $\mathcal{A}$ for Parallel VRF Game (defined in Lemma 4.4.1), as follows. $\mathcal{A}$ first invokes $\mathsf{R\text{-}RS.Gen}(1^k)$ to obtain (vk, sk). The vk is a tuple $(\vec{vk}_{\mathsf{VRF}}, \rho, \vec{\alpha})$, where $\vec{vk}_{\mathsf{VRF}}$ can be parsed further as $(vk^1_{\mathsf{VRF}}, \dots, vk^4_{\mathsf{VRF}})$.

$\mathcal{A}$ obtains two verification keys $vk^{3,*}_{\mathsf{VRF}}, vk^{4,*}_{\mathsf{VRF}}$ from the VRF challenger, and replaces vk^3_{VRF} and vk^4_{VRF} with these keys, setting

$$vk' = ((vk^1_{\mathsf{VRF}}, vk^2_{\mathsf{VRF}}, vk^{3,*}_{\mathsf{VRF}}, vk^{4,*}_{\mathsf{VRF}}), \rho, \vec{\alpha}).$$

Let $sk_{\mathsf{VRF}}^{3,*}, sk_{\mathsf{VRF}}^{4,*}$ be the VRF secret keys corresponding to $vk_{\mathsf{VRF}}^{3,*}, vk_{\mathsf{VRF}}^{4,*}$, respectively.[15]

Next, $\mathcal{A}$ runs $\mathcal{A}_{\mathsf{Sign}}(vk')$, answering $\mathcal{A}_{\mathsf{Sign}}$'s oracle queries as follows.

- On query (m'', R'') to OSign: $\mathcal{A}$ runs the honest signing algorithm $\mathsf{R\text{-}RS.Sign}$ on input $(R'' \cup \{vk'\}, sk, m'')$, with the following modification: in step 6, instead of using sk_{VRF}^{3} and sk_{VRF}^{4} to generate y_3, τ_3, and y_4, τ_4, $\mathcal{A}$ invokes its VRF oracle.

- On query (σ'', R'') to ORpd: $\mathcal{A}$ runs the honest repudiation algorithm $\mathsf{R\text{-}RS.Repudiate}$ on input $(R'' \cup \{vk'\}, sk, \sigma'')$. (Note that sk_{VRF}^{3} and sk_{VRF}^{4} are not used in algorithm $\mathsf{R\text{-}RS.Repudiate}$, so we don't need to invoke the VRF oracle here.)

Let (σ, m, R) be the output of $\mathcal{A}_{\mathsf{Sign}}$. $\mathcal{A}$ parses $\sigma = (\vec{\pi}, \vec{y}, \varphi)$ and $\vec{y} = (y_1, \ldots, y_4)$. Then $\mathcal{A}$ sends (R, m, φ) to the VRF challenger, receiving responses y_3' and y_4'. If $y_3 = y_3'$ or $y_4 = y_4'$, $\mathcal{A}$ outputs 0. Otherwise, $\mathcal{A}$ outputs a random bit. Let us now consider $\mathcal{A}$'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 $\mathcal{A}_{\mathsf{Sign}}$ is identical to the view in (4.12), so by assumption $\mathcal{A}_{\mathsf{Sign}}$ 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 $\mathcal{A}_{\mathsf{Sign}}$ wins the game, the condition $Q \cap \{(\cdot, m, R')\} \neq \varnothing$ in (4.12) implies that $\mathcal{A}$ has not previously made an oracle query on the VRF challenge message (R, m, φ) during the query phase. Let us suppose that $\mathcal{A}_{\mathsf{Sign}}$ wins the game described in (4.12), and consider the implications.

By definition, if $\mathsf{R\text{-}RS.VerRepud}$ rejects (with non-negligible probability) on an honestly generated repudiation $\xi = (\xi_i)_{i \in [|R|]}$ generated with respect to vk', then

$$\exists i \in [|R|] \text{ s.t. } \mathsf{ZAP.Verify}_{L'}(\rho_i, \xi_i, (R, m, \varphi, \vec{y}, vk')) = 0 \; . \tag{4.13}$$

By the completeness of the ZAP and the complete provability of the VRF, since ξ is honestly generated with respect to vk', the statement $\neg b_3' \vee \neg b_4'$ then holds with overwhelming

[15]Note that $sk_{\mathsf{VRF}}^{3,*}, sk_{\mathsf{VRF}}^{4,*}$ are generated by the VRF challenger and not accessible by $\mathcal{A}$.

probability, where b'_3, b'_4 are as defined in Definition 4.4.5. Expanding the definition of b'_3, b'_4, and again using that ξ is honestly generated, we have that

$$\begin{gathered}\exists \eta \in \{3,4\}, i' \in [|R|], j' \in [M], \gamma \text{ s.t.} \\ \mathsf{VRF.Verify}(vk^{\eta,*}_{\mathsf{VRF}}, (R,m,\varphi), y_\eta, \mathsf{VRF.Prove}(sk^{\eta,*}_{\mathsf{VRF}}, (R,m,\varphi)); \alpha^{i'}_{j'} \oplus \gamma) = 0 \ .\end{gathered} \tag{4.14}$$

Since $vk' \in R$ is honestly generated by assumption, we have that the $\vec{\alpha} = (\alpha_1, \ldots, \alpha_M)$ within vk' is distributed uniformly over $(\{0,1\}^\nu)^M$. Then applying Corollary 4.4.6 (setting the algorithm $\mathcal{V}$ to be $\mathsf{VRF.Verify}$): (4.14) implies *either*

$$\begin{gathered}\exists \eta \in \{3,4\} \text{ and } \tau \text{ s.t.} \\ \Pr\left[\mathsf{VRF.Verify}(vk^{\eta,*}_{\mathsf{VRF}}, (R,m,\varphi), y_\eta, \tau) = 1\right] \text{ is overwhelming,}\end{gathered} \tag{4.15}$$

or a negligible probability event occurred. By the complete and unique provability of the VRF, (4.15) implies that

$$y_3 = \mathsf{VRF.Eval}(sk^{3,*}_{\mathsf{VRF}}, (R,m,\varphi)) \text{ or } y_4 = \mathsf{VRF.Eval}(sk^{4,*}_{\mathsf{VRF}}, (R,m,\varphi)) \ . \tag{4.16}$$

Chaining together the implications, we conclude that (4.16) holds with all but negligible probability conditioned on the non-negligible-probability event of $\mathcal{A}_{\mathsf{Sign}}$ winning the game described in (4.12).

Finally, by definition of Parallel VRF Game, when $b = 0$,

$$y'_3 = \mathsf{VRF.Eval}(sk^{3,*}_{\mathsf{VRF}}, (R,m,\varphi)) \text{ and } y'_4 = \mathsf{VRF.Eval}(sk^{4,*}_{\mathsf{VRF}}, (R,m,\varphi)) \ . \tag{4.17}$$

From (4.16) and (4.17): when $b = 0$, there is a non-negligible probability that

$$y_3 = y'_3 \text{ or } y_4 = y'_4 \ . \tag{4.18}$$

Recall that (4.18) is the trigger condition for $\mathcal{A}$ to output 0. Therefore, when $b = 0$, $\mathcal{A}$ 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 $\mathcal{A}$ outputs a random bit.

Thus, $\mathcal{A}$ wins Parallel VRF Game with non-negligible probability. This contradicts the security of the VRF. Therefore, R-RS satisfies (4.3).

It remains to show that R-RS satisfies (4.4). The argument for this part of the proof follows a somewhat similar outline to the argument already presented. The rest of the proof is deferred to Section 4.8. $\square$

Lemma 4.4.8 (Unforgeability of R-RS). R-RS *is unforgeable (in the sense of Definition 4.3.5).*

Proof. This proof is very similar to the second half of the proof of Lemma 4.4.7.

Suppose that this is not the case. Then there exists some $N = \mathsf{poly}(k)$ and some PPT $\mathcal{B}$ 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:

$$\Pr\left[\begin{array}{l} (vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k) \\ (R^*, m^*, \sigma^*) \leftarrow \mathcal{B}^{\mathsf{OSign},\mathsf{ORpd},\mathsf{Corr}}(vk_1, \ldots, vk_N) \\ b \leftarrow \mathsf{Verify}(R^*, \sigma^*, m^*) \end{array} : \begin{array}{l} b = 1 \wedge R^* \subseteq \{vk_1, \ldots, vk_N\} \setminus I \\ \wedge Q \cap \{(\cdot, m^*, R^*)\} = \varnothing \end{array}\right]. \tag{4.19}$$

We build an adversary $\mathcal{A}$ to the Parallel VRF Game of Lemma 4.4.1. $\mathcal{A}$ first samples

$$(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k)$$

and parses each vk_i and sk_i as follows:

$$vk_i = (\vec{vk}^{\,i}_{\mathsf{VRF}} = (vk^{i,1}_{\mathsf{VRF}}, vk^{i,2}_{\mathsf{VRF}}, vk^{i,3}_{\mathsf{VRF}}, vk^{i,4}_{\mathsf{VRF}}), \rho_i, \vec{\alpha}_i)$$
$$sk_i = (\vec{sk}^{\,i}_{\mathsf{VRF}} = (sk^{i,1}_{\mathsf{VRF}}, sk^{i,2}_{\mathsf{VRF}}, sk^{i,3}_{\mathsf{VRF}}, sk^{i,4}_{\mathsf{VRF}}), vk_i)$$

Then $\mathcal{A}$ chooses a random $i^* \leftarrow [N]$, obtains a pair of verification keys $vk_{\mathsf{VRF}}^{*,3}, vk_{\mathsf{VRF}}^{*,4}$ from the VRF challenger, and lets

$$vk^* = (\vec{vk}_{\mathsf{VRF}}^{i} = (vk_{\mathsf{VRF}}^{i,1}, vk_{\mathsf{VRF}}^{i,2}, vk_{\mathsf{VRF}}^{*,3}, vk_{\mathsf{VRF}}^{*,4}), \rho_i, \vec{\alpha}_i).$$

Let $sk_{\mathsf{VRF}}^{*,3}, sk_{\mathsf{VRF}}^{*,4}$ denote the VRF secret keys corresponding to $vk_{\mathsf{VRF}}^{*,3}, vk_{\mathsf{VRF}}^{*,4}$, respectively.

Let $vk_{i^*}^* = vk^*$ and let $vk_i^* = vk_i$ for every $i \neq i^*$. Let $R^* = \{vk_1^*, \ldots, vk_N^*\}$. $\mathcal{A}$ then runs $\mathcal{B}$ on input R^*, answering $\mathcal{B}$'s oracle queries as follows.

- On query (i'', m'', R'') to OSign: $\mathcal{A}$ runs the honest signing algorithm $\mathsf{R\text{-}RS.Sign}$ on input $(R'' \cup \{vk_{i''}^*\}, sk_{i''}, m'')$, with the following modification if $i = i^*$: in step 6, instead of using $sk_{\mathsf{VRF}}^{i^*,3}$ and $sk_{\mathsf{VRF}}^{i^*,4}$ to generate y_3, τ_3, and y_4, τ_4, $\mathcal{A}$ invokes its VRF oracle.

- On query (i'', σ'', R'') to ORpd: $\mathcal{A}$ runs the honest repudiation algorithm $\mathsf{R\text{-}RS.Repudiate}$ on input $(R'' \cup \{vk_{i''}^*\}, sk_{i''}, \sigma'')$. ($sk_{\mathsf{VRF}}^3$ and sk_{VRF}^4 are not used by $\mathsf{R\text{-}RS.Repudiate}$, so $\mathcal{A}$ does not need to invoke the VRF oracle here.)

- On each invocation of Corr on some index $i \in [N]$: if $i = i^*$ then $\mathcal{A}$ outputs a random bit and aborts; otherwise, $\mathcal{A}$ responds to $\mathcal{B}$ with sk_i.

Let (R', m, σ) be the output of $\mathcal{B}$. $\mathcal{A}$ parses $\sigma = (\vec{\pi}, \vec{y}, \varphi)$ and $\vec{y} = (y_1, \ldots, y_4)$, and submits (R', m, φ) to the VRF challenger and then receive responses y_3' and y_4'. If $y_3' = y_3$ or $y_4' = y_4$, $\mathcal{A}$ outputs 0. Else, $\mathcal{A}$ outputs a random bit.

It remains to show that $\mathcal{A}$ distinguishes with non-negligible advantage, in the parallel VRF security game, between VRF outputs and random values.

Let us consider the behavior of $\mathcal{A}$ 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 $\mathcal{A}$ receives to the challenge (R', m) consists of VRF outputs on input (R', m) with respect to the keys $vk_{\mathsf{VRF}}^{*,3}, vk_{\mathsf{VRF}}^{*,4}$. In particular, whenever $\mathcal{B}$ does not query Corr on i^*, the view of $\mathcal{B}$ is identical to the view in (4.19). So, conditioned on $\mathcal{B}$ not corrupting i^*, $\mathcal{B}$ will win the game — i.e., produce a valid signature — with non-negligible probability, by assumption.

Let us consider the probability that $\mathcal{B}$ queries Corr for input i^*. Recall that this event causes $\mathcal{A}$

to abort and output a random bit. The distribution of the view (i.e., verification keys and oracle responses) of $\mathcal{B}$ is unaffected by $\mathcal{A}$'s choice of i^*, until the point at which $\mathcal{B}$ submits an oracle query to Corr for input i^* (if at all). The condition $R^* \subseteq \{vk_1, \ldots, vk_N\} \setminus I$ in (4.19) ensures that if $\mathcal{B}$ wins the game with non-negligible probability, then $\mathcal{B}$ leaves one or more keys uncorrupted with at least that non-negligible probability. Since i^* is chosen at random by $\mathcal{A}$, it follows that $\Pr[i^* \notin I]$ is non-negligible.

Let E' denote the event that $\mathcal{A}$ does not abort (i.e., $i^* \notin I$) and $\mathcal{B}$'s output signature verifies (i.e., $\mathsf{R\text{-}RS.Verify}(R', \sigma, m) = 1$). We have established that E' occurs with non-negligible probability. Then, by the same argument given from (4.28) to (4.32) in the proof of Lemma 4.4.7, we have that *either*

$$\exists j^* \in [|R'|], \eta \in \{3,4\}, \text{ and } \tau \text{ s.t.}$$
$$\Pr\left[\mathsf{VRF.Verify}(vk_{\mathsf{VRF}}^{j^*,\eta}, (R', m, \varphi), y_\eta, \tau) = 1\right] \text{ is overwhelming,}$$

or a negligible probability event occurred. When $j^* = i^*$, this moreover implies

$$y_3 = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{*,3}, (R', m, \varphi)) \text{ or } y_4 = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{*,3}, (R', m, \varphi)) \ . \tag{4.20}$$

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 $\mathcal{B}$ is unaffected by $\mathcal{A}$'s choice of i^*, until the point at which $\mathcal{B}$ submits an oracle query to Corr for input i^* (if at all). Since i^* is chosen at random by $\mathcal{A}$ (and is thus independent of j^*), and $i^*, j^* \in [|R'|]$, $\Pr[i^* = j^*]$ must be non-negligible. Therefore, (4.20) holds with non-negligible probability.

Finally, by definition of the Parallel VRF Game, whenever the challenger's bit b is 0,

$$y_3' = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{*,3}, (R', m, \varphi)) \text{ or } y_4' = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{*,3}, (R', m, \varphi)) \ . \tag{4.21}$$

From (4.20) and (4.21) we have that with non-negligible probability,

$$y_3 = y_3' \text{ or } y_4 = y_4' \ . \tag{4.22}$$

Recall that (4.22) is the trigger condition for $\mathcal{A}$ to output 0. We conclude that when the VRF challenger's bit $b = 0$, the trigger condition for $\mathcal{A}$ to output 0 is met with non-negligible probability; and by construction, $\mathcal{A}$ 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 $\mathcal{A}$ receives to the challenge message (R, m, φ) consists of truly random strings instead of VRF outputs, and so $\Pr[y_3' = y_3 \vee y_4' = y_4]$ is negligible. Thus, $\mathcal{A}$ outputs a random bit with overwhelming probability.

We have shown that $\mathcal{A}$'s output is non-negligibly different depending on the VRF challenger's bit, and so $\mathcal{A}$ must win the Parallel VRF Game with non-negligible probability. This contradicts the security of the VRF. Therefore, R-RS is unforgeable. □

Lemma 4.4.9 (Anonymity of R-RS)**.** R-RS *satisfies adaptive anonymity against adversarially chosen keys (Definition 4.3.4).*

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_1^* rather than i_0^*. 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^*. Details of the proof are deferred to Section 4.9. □

4.5 Claimable transformation

In this section, we give a simple 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.*

4.5.1 Building blocks

We assume familiarity with the standard notions of commitment schemes, standard signatures, and PRFs, and simply establish syntax in this subsection.[16] For simplicity, we denote a

[16]We refer to any standard textbook (e.g., [KL14]) for the relevant security definitions.

commitment scheme by a single algorithm Com, and assume that the decommitment simply consists of the commitment randomness.

Definition 4.5.1 (Commitment scheme). *A* commitment scheme Com *has the following syntax:* Com *takes as input a message μ and randomness r and outputs a commitment c. Sometimes we leave r implicit and write* $\mathsf{Com}(\mu)$ *instead of* $\mathsf{Com}(\mu; r)$*. The* decommitment *of c is the randomness r.*

A commitment scheme must satisfy the following properties.

- Hiding: *For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\exists$ negligible ε s.t. $\forall k \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{l} (\mu_0, \mu_1, \mathfrak{s}) \leftarrow \mathcal{A}_1(1^k) \\ b \leftarrow \{0,1\} \\ c \leftarrow \mathsf{Com}(\mu_b) \\ b' \leftarrow \mathcal{A}_2(c, \mathfrak{s}) \end{array} \quad : \quad b' = b \right] \leq 1/2 + \varepsilon(k) \ . \tag{4.23}$$

- Binding: *For all PPT adversaries $\mathcal{A}$, $\exists$ negligible ε s.t. $\forall k \in \mathbb{N}$,*

$$\Pr\Big[\ (c, \mu, r, \mu', r') \leftarrow \mathcal{A}(1^k) \quad : \quad \mu \neq \mu' \wedge \mathsf{Com}(\mu; r) = c = \mathsf{Com}(\mu'; r')\ \Big] \leq \varepsilon(k) \ .$$

Definition 4.5.2 (Standard signature (syntax)). *A* signature scheme *is a triple of PPT algorithms* $\Sigma = (\Sigma.\mathsf{Gen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ *with the following syntax:*

- $\Sigma.\mathsf{Gen}(1^k)$ *takes as input the security parameter k and outputs a* verification key vk *and a* signing key sk.

- $\Sigma.\mathsf{Sign}(sk, m)$ *takes as input a signing key sk and a message m, and outputs a signature σ.*

- $\Sigma.\mathsf{Verify}(vk, \sigma, m)$ *takes as input a verification key vk, a signature σ, and a message m, and outputs a single bit indicating whether or not σ is a valid signature on m w.r.t. vk.*

Definition 4.5.3 (PRF (syntax)). *A* pseudorandom function (PRF) *is a pair of algorithms* $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$*, where:*

- $\mathsf{PRF.Gen}$ *is a PPT algorithm that takes as input* 1^k *and outputs a* PRF key sk_{PRF}*, and*
- $\mathsf{PRF.Eval}$ *is a polynomial-time deterministic algorithm that takes as input a PRF key* sk_{PRF} *and* $x \in \{0,1\}^*$ *and outputs a string* r*.*

For simplicity, we assume PRFs that take arbitrary-length inputs (i.e., $\{0,1\}^*$*).*

4.5.2 The transformation

Our transformation builds on any ring signature scheme, RS, to construct a claimable ring signature scheme $\mathsf{C\text{-}RS}$. The basic idea is to take a signature σ_{RS} under RS and append to it a *commitment* c to $(vk, \sigma_{\mathsf{RS}})$ where vk is the verification key of the signer. The verification algorithm simply checks whether σ_{RS} verifies. The claim consists of a decommitment revealing that c is a commitment to $(vk, \sigma_{\mathsf{RS}})$. 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 $(\sigma_{\mathsf{RS}}, vk')$ where vk' is not his own key but that of someone else in the ring? This ability would violate equation (4.6) of Definition 4.3.9 (claimability). To prevent such behavior, our construction actually commits to a *standard (non-ring) signature* on $(vk, \sigma_{\mathsf{RS}})$. 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 4.3.9 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 4.5.1. *Our transformation* $\mathsf{C\text{-}RS}$ *is parametrized by the following:*

- RS, *a ring signature scheme,*
- Σ, *a standard signature scheme,*
- Com, *a commitment scheme, and*
- PRF, *a PRF.*

For convenience, and without loss of generality, we assume that the commitment randomness of Com, *the signing randomness of* Σ, *and the output of* $\mathsf{PRF.Eval}$ *all have the same length of* ν *bits.*

$\mathsf{C\text{-}RS.Gen}(1^k)$

1. *Let* $(vk_{\mathsf{RS}}, sk_{\mathsf{RS}}) \leftarrow \mathsf{RS.Gen}(1^k)$.
2. *Let* $(vk_\Sigma, sk_\Sigma) \leftarrow \Sigma.\mathsf{Gen}(1^k)$.
3. *Let* $sk_{\mathsf{PRF}} \leftarrow \mathsf{PRF.Gen}(1^k)$.
4. *Output* $vk = (vk_{\mathsf{RS}}, vk_\Sigma)$ *and* $sk = (vk, sk_{\mathsf{RS}}, sk_\Sigma, sk_{\mathsf{PRF}})$.

In the rest of the construction, we implicitly parse verification keys and signing keys of $\mathsf{C\text{-}RS}$ *as* $vk = (vk_{\mathsf{RS}}, vk_\Sigma)$ *and* $sk = (vk, sk_{\mathsf{RS}}, sk_\Sigma, sk_{\mathsf{PRF}})$ *respectively. Also, for a ring*

$$R = \big(vk_1 = (vk^1_{\mathsf{RS}}, vk^1_\Sigma), \ldots, vk_N = (vk^N_{\mathsf{RS}}, vk^N_\Sigma)\big) \ ,$$

we write $\mathsf{RS}(R)$ *to denote* $(vk^1_{\mathsf{RS}}, \ldots, vk^N_{\mathsf{RS}})$.

$\mathsf{C\text{-}RS.Sign}(R, sk, m)$

1. *Let* $\sigma_{\mathsf{RS}} \leftarrow \mathsf{RS.Sign}(\mathsf{RS}(R), sk_{\mathsf{RS}}, m)$.
2. *Let* $r_\Sigma = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 0))$.

3. *Let* $\sigma_\Sigma = \Sigma.\mathsf{Sign}(sk_\Sigma, (vk, \sigma_{\mathsf{RS}}); r_\Sigma)$.

4. *Let* $r_{\mathsf{Com}} = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 1))$.

5. *Let* $c = \mathsf{Com}((vk, \sigma_\Sigma); r_{\mathsf{Com}})$.

6. *Let* $\sigma = (\sigma_{\mathsf{RS}}, c)$.

7. *If* $\mathsf{C\text{-}RS.VerClaim}(R, vk, \sigma, \mathsf{C\text{-}RS.Claim}(R, sk, \sigma)) = 1$, *output* σ.

8. *Otherwise, output* $(\perp, \perp)$.

$\mathsf{C\text{-}RS.Verify}(R, \sigma = (\sigma_{\mathsf{RS}}, c), m)$

1. *If* $\sigma_{\mathsf{RS}} = \perp$, *output* 0.

2. *Otherwise, output* $\mathsf{RS.Verify}(\mathsf{RS}(R), \sigma_{\mathsf{RS}}, m)$.

$\mathsf{C\text{-}RS.Claim}(R, sk, \sigma = (\sigma_{\mathsf{RS}}, c))$

1. *Let* $r'_\Sigma = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 0))$.

2. *Let* $r'_{\mathsf{Com}} = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 1))$.

3. *Let* $\sigma'_\Sigma = \Sigma.\mathsf{Sign}(sk_\Sigma, (vk, \sigma_{\mathsf{RS}}); r'_\Sigma)$.

4. *If* $c \neq \mathsf{Com}(\sigma'_\Sigma, r'_{\mathsf{Com}})$, *output* $\zeta = \perp$.

5. *Otherwise, output* $\zeta = (r'_{\mathsf{Com}}, \sigma'_\Sigma)$.

$\mathsf{C\text{-}RS.VerClaim}(R, vk, \sigma = (\sigma_{\mathsf{RS}}, c), \zeta = (r'_{\mathsf{Com}}, \sigma'_\Sigma))$

1. *Let* $c' = \mathsf{Com}((vk, \sigma'_\Sigma); r'_{\mathsf{Com}})$.

2. *Output* $(c = c') \wedge \Sigma.\mathsf{Verify}(vk_\Sigma, \sigma'_\Sigma, (vk, \sigma_{\mathsf{RS}}))$.

If RS *is a* repudiable *ring signature scheme equipped with algorithms* (Repudiate, VerRepud)*, then we additionally define* C-RS.Repudiate *and* C-RS.VerRepud *to simply run the* Repudiate *and* VerRepud *algorithms of* RS*, as follows.*

$\mathsf{C\text{-}RS.Repudiate}(R, sk, \sigma = (\sigma_{\mathsf{RS}}, c))$

1. *Output* $\mathsf{RS.Repudiate}(\mathsf{RS}(R), sk, \sigma_{\mathsf{RS}})$.

$\mathsf{C\text{-}RS.VerRepud}(R, vk, \sigma = (\sigma_{\mathsf{RS}}, c), \xi)$

1. *Output* $\mathsf{RS.VerRepud}(\mathsf{RS}(R), sk, \sigma_{\mathsf{RS}}, \xi)$.

Theorem 4.5.2. C-RS *is a claimable ring signature scheme (Theorem 4.5.3). Moreover, if* RS *is a repudiable ring signature scheme, then* C-RS *is repudiable-and-claimable (Theorem 4.5.8).*

Proof. Follows from Theorems 4.5.3 and 4.5.8. □

Theorem 4.5.3 (Claimability of C-RS). C-RS *is a claimable ring signature scheme.*

Proof. Follows from Lemmata 4.5.4–4.5.7, which establish the properties of correctness, claimability, unforgeability, and anonymity, respectively. □

Lemma 4.5.4 (Correctness of C-RS). C-RS *satisfies correctness (Definition 4.2.2).*

Correctness is immediate, so we omit the proof.

Lemma 4.5.5 (Claimability of C-RS). C-RS *is claimable (Definition 4.3.9).*

Proof. We show that C-RS satisfies each of the three conditions of Definition 4.3.9. The first condition is immediate by the correctness of the signature scheme Σ, since the use of the PRF ensures that the values $(r'_\Sigma, r'_{\mathsf{Com}}, \sigma'_\Sigma)$ computed in Claim are the same as the corresponding values computed in Sign and that the commitments c, c' match.

For the second condition, assume for contradiction that there exists some PPT malicious claiming algorithm $\mathcal{A}_{\mathsf{Claim}} = (\mathcal{A}_1, \mathcal{A}_2)$ that is able to claim a signature produced by a different party. That is, that the following probability is non-negligible:

$$\Pr\left[\begin{array}{lcl} (vk, sk) \leftarrow \mathsf{C\text{-}RS.Gen}(1^k) & & \\ (R', m) \leftarrow \mathcal{A}_1^{\mathcal{O}}(vk) & & \\ \sigma \leftarrow \mathsf{C\text{-}RS.Sign}(R' \cup \{vk\}, sk, m) & & b = 1 \wedge b' = 1 \\ (\zeta, vk') \leftarrow \mathcal{A}_2^{\mathcal{O}}(R' \cup \{vk\}, \sigma) & : & \wedge vk' \neq vk \\ b \leftarrow \mathsf{C\text{-}RS.VerClaim}(R' \cup \{vk\}, vk', \sigma, \zeta) & & \\ b' \leftarrow \mathsf{C\text{-}RS.Verify}(R' \cup \{vk\}, \sigma, m) & & \end{array}\right], \tag{4.24}$$

where $\mathcal{O} = \{\mathsf{OSign}, \mathsf{OClaim}_{vk,sk}\}$. We will produce an adversary $\mathcal{B}$ that breaks the binding property of the commitment scheme. Let $\mathcal{B}$ first run the experiment in Equation 4.24, invoking the malicious claiming algorithm $\mathcal{A}_{\mathsf{Claim}}$ and using its knowledge of the secret key sk to answer the oracle queries of $\mathcal{A}_{\mathsf{Claim}}$. Let $\mathcal{B}$ then compute $\zeta' = \mathsf{C\text{-}RS.Claim}(R' \cup \{vk\}, sk, \sigma)$ and $b'' = \mathsf{C\text{-}RS.VerClaim}(R' \cup \{vk\}, vk, \sigma, \zeta')$. Since the signature verifies correctly with non-negligible probability, the check on line 7 of $\mathsf{C\text{-}RS.Sign}$ must pass, and so we must have that $b'' = 1$ whenever $b' = 1$. Conseqently with non-negligible probability we have that $b = b' = b'' = 1$ and $vk \neq vk'$. Conditioning on this event, we have that $\sigma = (\sigma_{\mathsf{RS}}, c)$, $\zeta = (r_{\mathsf{Com}}, \sigma_\Sigma)$, and $\zeta' = (r'_{\mathsf{Com}}, \sigma'_\Sigma)$ are such that $c = \mathsf{Com}((vk, \sigma_\Sigma), r_{\mathsf{Com}}) = \mathsf{Com}((vk', \sigma'_\Sigma), r'_{\mathsf{Com}})$. But $vk \neq vk'$, so with non-negligible probability $\mathcal{B}$ 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 4.3.9.

For the third condition, assume for contradiction that there exists some PPT malicious signing-and-claiming algorithm $\mathcal{A}_{\mathsf{S\&C}}$ that produces a signature and claims it on behalf of a different party. That is, assume that the following probability is non-negligible:

$$\Pr\left[\begin{array}{lcl} (vk, sk) \leftarrow \mathsf{C\text{-}RS.Gen}(1^k) & & \\ (R', m, \sigma, \zeta) \leftarrow \mathcal{A}_{\mathsf{S\&C}}^{\mathcal{O}, \mathsf{OClaim}_{(vk,sk)}}(vk) & & b = 1 \wedge b' = 1 \\ b \leftarrow \mathsf{C\text{-}RS.VerClaim}(R' \cup \{vk\}, vk, \sigma, \zeta) & : & \wedge Q \cap \{(\cdot, \sigma)\} = \varnothing \\ b' \leftarrow \mathsf{C\text{-}RS.Verify}(R' \cup \{vk\}, \sigma, m) & & \end{array}\right], \tag{4.25}$$

where $\mathcal{O} = \{\mathsf{OSign}\}$ and Q is the set of queries made to the oracle $\mathsf{OClaim}_{(vk,sk)}$. We will construct an adversary $\mathcal{B}$ that breaks the unforgeability property of the signature scheme Σ. $\mathcal{B}$ first requests a verification key vk^*_Σ from the challenger for Σ. It samples keys $(vk, sk) \leftarrow \mathsf{C\text{-}RS.Gen}(1^k)$ as in the beginning of the experiment in Equation 4.25, but replaces vk_Σ with vk^*_Σ and sk_Σ with $\perp$ in vk and sk, respectively. It then invokes the adversary $\mathcal{A}_{\mathsf{S\&C}}$ on inputs $(1^k, vk)$ to produce values (R', m, σ, ζ), using the challenger for the signature scheme Σ to respond to the oracle queries of $\mathcal{A}_{\mathsf{S\&C}}$ as follows:

- When $\mathcal{A}_{\mathsf{S\&C}}$ queries oracle OSign on input (m, R), algorithm $\mathcal{B}$ first computes σ_{RS}

as in $\mathsf{C\text{-}RS.Sign}$. It then invokes the challenger for Σ on message $(vk, \sigma_{\mathsf{RS}})$, receiving signature σ_Σ. (If the challenger has already been queried on this message $(vk, \sigma_{\mathsf{RS}})$ in a previous invocation of OSign, then instead of querying the challenger again, use the same value σ_Σ sent by the challenger in the previous invocation.) It then proceeds as in steps 4–8 of algorithm $\mathsf{C\text{-}RS.Sign}$ computing value r_{Com} and commitment c, testing whether $\mathsf{C\text{-}RS.VerClaim}$ succeeds (where in the inner invocation of $\mathsf{C\text{-}RS.Claim}$ we set σ'_Σ to be the challenger-produced signature σ_Σ instead of running $\Sigma.\mathsf{Sign}$), and returning $\sigma = (\sigma_{\mathsf{RS}}, c)$.

- When $\mathcal{A}_{\mathsf{S\&C}}$ queries oracle $\mathsf{OClaim}_{(vk,sk)}$ on input (R, σ), algorithm $\mathcal{B}$ first tests whether σ was an output returned by a previous invocation of the oracle OSign. If not, it immediately returns $\perp$. If it is, then let σ_Σ be the signature returned by the challenger for Σ on that invocation of OSign. Compute value r_{Com} as in algorithm $\mathsf{C\text{-}RS.Claim}$, and output $\zeta = (r_{\mathsf{Com}}, \sigma_\Sigma)$.

Finally, algorithm $\mathcal{B}$ parses $\zeta = (r'_{\mathsf{Com}}, \sigma'_\Sigma)$ and outputs σ'_Σ as its forgery.

We now outline a hybrid argument to show that this adversary $\mathcal{B}$ breaks the unforgeability property of the signature scheme.

Hybrid 1. *The experiment with adversary $\mathcal{B}$ as just described.*

Hybrid 2. *Instead of the signature scheme challenger using actual randomness to generate the signatures in the oracle queries to* OSign, *use pseudorandomness obtained by a PRF invocation* $r_\Sigma = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 0))$

Hybrid 3. *Instead of responding to both types of oracle queries as above, use actual invocations of* $\mathsf{C\text{-}RS.Sign}$ *and* $\mathsf{C\text{-}RS.Claim}$, *using the secret key sk_Σ known to the challenger for the unforgeability game.*

Indistinguishability of Hybrids 1 and 2 follows from the pseudorandomness of the PRF. For Hybrids 2 and 3, note first that the two experiments behave identically on invocations of the oracle OSign. It remains to consider invocations of the oracle OClaim. By the binding property of the commitment scheme and the pseudorandomness of the PRF, with all but negligible probability adversary $\mathcal{A}_{\mathsf{S\&C}}$ in Hybrid 3 will be unable to find an input to the oracle OClaim that does not yield output $\perp$, except for inputs that were previously produced as output to the oracle OSign. Consequently, except with negligible probability, the oracle OClaim will behave identically in Hybrids 2 and 3. But Hybrid 3 is exactly the experiment in Equation 4.25, so with non-negligible probability adversary $\mathcal{A}_{\mathsf{S\&C}}$ in this experiment produces a signature $\sigma = (\sigma_{\mathsf{RS}}, c)$ and claim $\zeta = (r'_{\mathsf{Com}}, \sigma'_{\Sigma})$ such that σ was not a query to oracle $\mathsf{OClaim}_{(vk,sk)}$ and $\Sigma.\mathsf{Verify}(vk_{\Sigma}, \sigma'_{\Sigma}, (vk, \sigma_{\mathsf{RS}}))$ outputs 1, i.e. σ'_{Σ} verifies as a valid signature of the message $(vk, \sigma_{\mathsf{RS}})$ under key vk_{Σ}. By the hybrid argument, it follows that with non-negligible probability, $\mathcal{B}$ successfully produces a valid signature for message $(vk, \sigma_{\mathsf{RS}})$.

It remains to argue that this signature is a valid forgery, i.e. that its message is distinct from each of the signatures produced by the challenger. To achieve this, we will make a small modification to the adversary $\mathcal{B}$; call the new adversary $\mathcal{B}'$. Let $L = L(k)$ be a (polynomial) upper bound on the number of queries made by $\mathcal{B}$ to the oracle OSign, and choose at random an index $i^* \in [L]$ On the i^*th query to OSign, rather than invoking the ring signature challenger to obtain σ_{Σ} and computing commitment c, instead choose a random string σ^* and compute the commitment with respect to that. By the hiding property of the commitment scheme, as long as the oracle OClaim is not queried on this signature, the output of the modified adversary $\mathcal{B}'$ is indistinguishable from that of $\mathcal{B}$. We already have by assumption that with non-negligible probability $\mathcal{A}_{\mathsf{S\&C}}$ wins the experiment in Equation 4.25, but moreover we have with non-negligible probability that in addition either the signature σ produced by $\mathcal{A}_{\mathsf{S\&C}}$ was never the output of oracle OSign (and that i^* is greater than the number of queries made to oracle OSign) or that it was the output of the i^*th query to OSign (and no earlier query). In the latter event, since $\mathcal{A}_{\mathsf{S\&C}}$ wins the experiment in Equation 4.25, this signature cannot have been a query to oracle OClaim, and so in either case, the behavior of $\mathcal{B}'$ is indistinguishable from $\mathcal{B}$. But in either case, the signature σ in the experiment with $\mathcal{B}'$ was not produced by invoking the challenger to the signature scheme, so it follows that (except with negligible probability), the signature σ'_{Σ} from the claim ζ was not produced by the signature scheme challenger. Putting everything together, we have that with non-negligible probability, the adversary $\mathcal{A}_{\mathsf{S\&C}}$ in the experiment with $\mathcal{B}'$ produces a valid signature for a message distinct from any message signed by the signature scheme challenger. This violates the unforgeability property of the signature scheme and yields a contradiction, and so the third property of Definition 4.3.9 is also satisfied. □

Lemma 4.5.6 (Unforgeability of C-RS). C-RS *is unforgeable (in the sense of Definition 4.3.11).*

Proof. The proof is by reduction to the unforgeability of RS.[17] Suppose, for contradiction, that there is a PPT adversary $\mathcal{A}$ that violates unforgeability of C-RS (Definition 4.3.11). Then we construct another adversary $\mathcal{B}$ that violates unforgeability of RS *without having access to a* OClaim *oracle.* On input $(vk_1, \ldots, vk_N)$ which are verification keys of RS, $\mathcal{B}$ behaves as follows.

1. For each $i \in [N]$:
   - Sample $(vk^i_\Sigma, sk^i_\Sigma) \leftarrow \Sigma.\mathsf{Gen}(1^k)$.
   - Sample $sk^i_{\mathsf{PRF}} \leftarrow \mathsf{PRF.Gen}(1^k)$.
   - Let $vk^*_i = (vk_i, vk^i_\Sigma)$ and $sk^*_i = (sk_i, sk^i_\Sigma, sk^i_{\mathsf{PRF}})$.
2. Run $\mathcal{A}$ on input $(vk^*_1, \ldots, vk^*_N)$, answering $\mathcal{A}$'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 σ_{RS}.
      - Let $r_\Sigma = \mathsf{PRF.Eval}(sk^i_{\mathsf{PRF}}, (vk^*_i, \sigma_{\mathsf{RS}}, 0))$.
      - Let $\sigma_\Sigma = \Sigma.\mathsf{Sign}(sk^i_\Sigma, (vk^*_i, \sigma_{\mathsf{RS}}); r_\Sigma)$.
      - Let $r_{\mathsf{Com}} = \mathsf{PRF.Eval}(sk^i_{\mathsf{PRF}}, (vk^*_i, \sigma_{\mathsf{RS}}, 1))$.
      - Let $c = \mathsf{Com}(\sigma_\Sigma; r_{\mathsf{Com}})$.

[17]Note that the unforgeability guarantee we have on RS is standard unforgeability of ring signatures (Definition 4.2.8), which does not give the adversary a OClaim oracle. If we had the stronger guarantee that RS were unforgeable in the presence of a OClaim oracle, then the unforgeability of C-RS would follow immediately, since signatures of C-RS contain signatures of RS.

- Output $\sigma = (\sigma_{\mathsf{RS}}, c)$.

(b) For each query (i, R, σ) to $\mathsf{C\text{-}RS.OClaim}$:

- Parse σ as $(\sigma_{\mathsf{RS}}, c)$.
- Let $r'_\Sigma = \mathsf{PRF.Eval}(sk^i_{\mathsf{PRF}}, (vk^*_i, \sigma_{\mathsf{RS}}, 0))$.
- Let $r'_{\mathsf{Com}} = \mathsf{PRF.Eval}(sk^i_{\mathsf{PRF}}, (vk^*_i, \sigma_{\mathsf{RS}}, 1))$.
- Let $\sigma'_\Sigma = \Sigma.\mathsf{Sign}(sk^i_\Sigma, (vk^i_\Sigma, \sigma_{\mathsf{RS}}); r'_\Sigma)$.
- Output $\zeta = (r'_{\mathsf{Com}}, \sigma'_\Sigma)$.

3. Upon receiving an output (R', m', σ') from $\mathcal{A}$: parse σ' as $(\sigma'_{\mathsf{RS}}, c')$, define $R'' = \{vk_i : vk^*_i \in R'\}$, and output $(R'', m', \sigma'_{\mathsf{RS}})$.

By construction of C-RS and $\mathcal{B}$, whenever $\mathcal{A}$ successfully forges with respect to C-RS, $\mathcal{B}$ successfully forges with respect to RS. Moreover, $\mathcal{B}$'s responses to $\mathcal{A}$'s oracle queries are, by construction, distributed identically to the oracle responses in the unforgeability experiment for C-RS. Therefore, $\mathcal{A}$'s probability of successful forgery is the same when $\mathcal{B}$ runs $\mathcal{A}$ in the above reduction, as in the unforgeability experiment. By our supposition, $\mathcal{A}$'s forging probability in the unforgeability experiment is some non-negligible ε, so it follows that $\mathcal{A}$'s forging probability in the above reduction is also ε, and therefore $\mathcal{B}$'s forging probability with respect to RS is in turn ε. This contradicts the unforgeability of RS. The lemma follows. ☐

Lemma 4.5.7 (Anonymity of C-RS). *If* RS *satisfies anonymity (resp., adaptive anonymity) against adversarially chosen keys (Definition 4.2.5), then* C-RS *satisfies anonymity (resp., adaptive anonymity) against adversarially chosen keys (Definition 4.3.10).*

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 4.2.3) for anonymity against adversarially chosen keys (Definition 4.3.10), shown below as "Hybrid 0."

Anonymity experiment (Hybrid 0)

```
$(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k)$
$((m^*, i_0^*, i_1^*, R^*), \mathfrak{s}) \leftarrow \mathcal{A}_1^{\mathsf{OSign},\mathsf{OClaim},\mathsf{Corr}}(vk_1, \ldots, vk_N)$
$b \leftarrow \{0,1\}$
$\sigma \leftarrow \mathsf{Sign}(R^* \cup \{vk_{i_0^*}, vk_{i_1^*}\}, sk_{i_b^*}, m^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{OSign},\mathsf{OClaim}^{\langle\sigma\rangle},\mathsf{Corr}}(\mathfrak{s}, \sigma)$
```

We now define two signing algorithms Sign_1 and Sign_2 which are slight variants of C-RS.Sign. For $\iota \in \{1,2\}$, we define Hybrid ι to be the same as Hybrid 0 except that the invocation of Sign in the fourth line of the experiment is replaced by an invocation of Sign_ι. In the descriptions of Sign_1 and 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.

$\mathsf{Sign}_1(R, sk, m)$

```
1. Let $\sigma_{\mathsf{RS}} \leftarrow \mathsf{RS.Sign}(\mathsf{RS}(R), sk_{\mathsf{RS}}, m)$.
2. Let $r_\Sigma = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 0))$.
3. Let $\sigma_\Sigma = \Sigma.\mathsf{Sign}(sk_\Sigma, (vk, \sigma_{\mathsf{RS}}); r_\Sigma)$.
4. Let $r_{\mathsf{Com}} \leftarrow \{0,1\}^\nu$.
5. Let $c = \mathsf{Com}(\sigma_\Sigma; r_{\mathsf{Com}})$.
6. Output $\sigma = (\sigma_{\mathsf{RS}}, c)$.
```

$\mathsf{Sign}_2(R, sk, m)$

```
1. Let $\sigma_{\mathsf{RS}} \leftarrow \mathsf{RS.Sign}(\mathsf{RS}(R), sk_{\mathsf{RS}}, m)$.
2. ~~Let $r_\Sigma = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 0))$.~~
3. ~~Let $\sigma_\Sigma = \Sigma.\mathsf{Sign}(sk_\Sigma, (vk, \sigma_{\mathsf{RS}}); r_\Sigma)$.~~
4. Let $r_{\mathsf{Com}} \leftarrow \{0,1\}^\nu$.
5. Let $c = \mathsf{Com}(0; r_{\mathsf{Com}})$.
6. Output $\sigma = (\sigma_{\mathsf{RS}}, c)$.
```

Hybrid 1 is indistinguishable from Hybrid 0. This follows from PRF security as long as there are no other variables in $\mathcal{A}$'s view that are correlated with the PRF output in Hybrid 0, namely, $r_{\mathsf{Com}} = \mathsf{PRF.Eval}(sk_{\mathsf{PRF}}, (vk, \sigma_{\mathsf{RS}}, 1))$. 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 $(vk, \sigma_{\mathsf{RS}}, 1)$. The only PRF evaluations during the experiment are to compute the r_{Σ} and r_{Com} values used by the OSign oracle when responding to oracle queries. The PRF inputs used to compute r_{Σ} values are distinct, by construction, by from those used to compute r_{Com} values (since the former end in 0 and the latter end in 1). The PRF inputs used to compute r_{Com} values for OSign queries are also, with overwhelming probability, distinct from the challenge input $(vk, \sigma_{\mathsf{RS}}, 1)$. Since each such PRF input is of the format $(vk', \sigma'_{\mathsf{RS}}, 1)$ where σ'_{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[18] — key pair.

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

[18]With knowledge of vk but not sk.

(Definition 4.2.8), which does not give the adversary a OClaim oracle.

Suppose, for contradiction, that there is a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that violates anonymity of $\mathsf{C\text{-}RS}$ (Definition 4.3.10). Then we construct another adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that violates anonymity of RS *without having access to a* OClaim *oracle.* On input $(vk_1, \dots, vk_N)$ which are verification keys of RS, $\mathcal{B}_1$ behaves as follows.

1. For each $i \in [N]$, construct $vk_i^* = (vk_i, vk_\Sigma^i)$ and $sk_i^* = (sk_i, sk_\Sigma^i, sk_{\mathsf{PRF}}^i)$ exactly as described in Step 1 in the proof of Lemma 4.5.6.

2. Run $\mathcal{A}$ on input $(vk_1^*, \dots, vk_N^*)$, answering $\mathcal{A}$'s oracle queries exactly as described in Step 2 in the proof of Lemma 4.5.6.

3. Upon receiving an output $((m', i_0', i_1', R'), \mathfrak{s}')$ from $\mathcal{A}$: define $R'' = \{vk_i : vk_i^* \in R'\}$, let $\mathfrak{s}''$ be all of $\mathcal{B}$'s internal state, and output $((m', i_0', i_1', R''), \mathfrak{s}'')$.

Then, on input $(\mathfrak{s}'', \sigma_{\mathsf{RS}})$, $\mathcal{B}_2$ behaves as follows.

1. Let $c = \mathsf{Com}(0; r)$ for truly random r.

2. Let $\sigma = (\sigma_{\mathsf{RS}}, c)$.

3. Run $\mathcal{A}_2$ to obtain $b' \leftarrow \mathcal{A}_2(\mathfrak{s}', \sigma)$.

4. Output b'.

$\mathcal{A}$'s view between the reduction run by $\mathcal{B}$ is identically distributed to $\mathcal{A}$'s view in the experiment of Hybrid 2. Moreover, by construction, $\mathcal{B}_2$'s guess is correct exactly when $\mathcal{A}_2$ guesses b' correctly. Therefore, $\mathcal{B}$'s success probability in the anonymity experiment of RS is negligibly close to $\mathcal{A}$'s success probability in the anonymity experiment of $\mathsf{C\text{-}RS}$. By supposition, the latter probability is non-negligibly greater than $1/2$. It follows that $\mathcal{B}$ violates the anonymity of RS, which is contradiction. The lemma follows. □

Theorem 4.5.8 (Repudiability-and-claimability of $\mathsf{C\text{-}RS}$). *If* RS *is repudiable, then* $\mathsf{C\text{-}RS}$ *is a repudiable-and-claimable ring signature scheme. (Definition 4.7.1).*

Proof (sketch). Suppose RS is repudiable. We need to prove that $\mathsf{C\text{-}RS}$ satisfies the definitions (which are given in Appendix 4.7) of repudiability, claimability, anonymity, and unforgeability of repudiable-and-claimable ring signature schemes.

The proofs of anonymity and unforgeability are essentially identical to the proofs of Lemmata 4.5.7 and 4.5.6, respectively. Those proofs reduce the anonymity/unforgeability of $\mathsf{C\text{-}RS}$ to the anonymity/unforgeability of RS (as defined in Section 4.2), respectively, under the assumption that RS is a *standard* (i.e., not necessarily repudiable) ring signature scheme. The same proof structure suffices to argue that in the case that RS is a repudiable ring signature scheme, that the anonymity/unforgeability of $\mathsf{C\text{-}RS}$ reduces to the anonymity/unforgeability notions for *repudiable* ring signature schemes (as defined in Section 4.3.1), which by assumption are satisfed by RS.

It remains to prove repudiability and claimability of $\mathsf{C\text{-}RS}$, according to Definitions 4.7.2 and 4.7.3, respectively.

Repudiability We reduce the repudiability of $\mathsf{C\text{-}RS}$ to the repudiability of RS. Suppose, for contradiction, that $\mathsf{C\text{-}RS}$ did not satisfy Definition 4.7.2. Then the repudiability of RS could be violated by an adversary $\mathcal{A}$ that instantiates its own commitment and signature schemes and PRF, and runs the adversary $\mathcal{B}$ that breaks the repudiability of $\mathsf{C\text{-}RS}$, while:

- augmenting each signature under RS with a corresponding commitment so that it appears indistinguishable from a signature under $\mathsf{C\text{-}RS}$ in $\mathcal{B}$'s view; and
- remembering the commitment randomness for any such commitments; and
- answering oracle calls to OClaim by producing the appropriate decommitments (using the remembered commitment randomness); and
- answering oracle calls to ORpd by passing them to its own ORpd oracle for RS.

Claimability The proof structure for claimability is very similar to that of Lemma 4.5.5, which argues that $\mathsf{C\text{-}RS}$ satisfies each of the three conditions of Definition 4.3.9 in turn.

- The argument for the first condition (i.e., that honest claims are validated by VerClaim) goes through unchanged.

- The argument for the second condition (i.e., that non-signers cannot successfully claim) also goes through unchanged: the adversary $\mathcal{B}$ constructed in the proof of Lemma 4.5.5 has knowledge of the secret key sk, which it can use to answer ORpd queries using the honest repudiation algorithm.

- The argument for the third condition needs to be augmented with a description of how $\mathcal{B}$ responds to oracle queries to ORpd that are made by $\mathcal{A}$. Just as above, $\mathcal{B}$ answers these oracle queries by running the honest repudiation algorithm. □

4.6 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 [BK10], augmented with an additional algorithm ExtractRandomness.

We first give a very brief summary of necessary background on lattice trapdoors; see [GPV08] and Appendix 4.10 for details.

4.6.1 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* ($\mathsf{SIS}_{q,m,\beta}$) 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 $\|x\| \le \beta$. For polynomial m', β and prime $q \ge \beta \cdot \omega(\sqrt{n \log n})$, 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 $\beta \cdot \tilde{O}(\sqrt{n})$ [MR07, GPV08].

Let $D_{\Lambda,s,c}$ denote the discrete Gaussian distribution over n-dimensional lattice Λ, centered at $c \in \mathbb{R}^n$ and with parameter s. We note the existence of the following algorithms, described in [GPV08][19]:

- 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 $\Lambda^{\perp}(A)$.
- There is an algorithm SampleDist sampling from the discrete Gaussian distribution $D_{\mathbb{Z}^{m'},s,0}$.
- There is an algorithm 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 $D_{\mathbb{Z}^{m'},s,0}$ 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 SampleDist and SampleCond, respectively, sample randomness under which the algorithm produces the desired output.

- There is an algorithm ExplainDist that on input an image vector x and parameter s, samples randomness ρ that yields output x under algorithm SampleDist, i.e. samples from the distribution $\{\rho | \mathsf{SampleDist}(s;\rho) = x\}$.
- There is an algorithm ExplainCond that on input matrix A, trapdoor T, parameter s, vector u and image vector x, samples randomness ρ that yields output x under algorithm SampleCond with inputs (A, T, s, u), i.e. samples from the distribution $\{\rho | \mathsf{SampleCond}(A, T, s, u; \rho) = x\}$.

We describe the algorithms ExplainDist and ExplainCond in Appendix 4.10. We will use a slight modification of the SampleCond algorithm of [GPV08] that uses the basis randomization technique of [CHKP10]. We need the following lemma.

[19]These are given by thealgorithms TrapGen, SampleD and SampleISIS in [GPV08].

Lemma 4.6.1. *Let (A_1, T_1) and (A_2, T_2) be sampled from* TrapdoorSamp*, let $y \in \mathbb{Z}_q^n$, and let $s \geq \max(\|\tilde{T_1}\|, \|\tilde{T_2}\|) \cdot \omega(\sqrt{\log n})$, where the tilde denotes Gram-Schmidt orthogonalization. Sample vectors x_1 and x_2' from* SampleDist*. Let $x_2 \leftarrow$* SampleCond*$(A_2, T_2, s, y - A_1x_1)$, and let $x_1' \leftarrow$* 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 follows immediately from Lemma 3.3 of [CHKP10].

4.6.2 The basic construction of [BK10]

We now describe the construction of [BK10].[20] 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.

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 of "short" vectors.

The key generation algorithm samples a matrix with an SIS trapdoor, and an additional set of 2ℓ 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.

```
BK-RS.Gen($1^k$)

    1. Let $(A, T) \leftarrow$ 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 $vk = (A, (A_{j,b})_{(j,b)\in[\ell]\times\{0,1\}}, y)$ and $sk = (vk, T)$.
```

[20]The original presentation of [BK10] introduces an abstraction they call *ring trapdoor functions*, and instantiates this abstraction from biliear 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.

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 4.26 being satisfied. The signature consists of the list of short vectors for each identity and each index of the message.

$\mathsf{BK\text{-}RS.Sign}(R, sk, m; \rho)$

1. Parse $R = (vk_1, \ldots, vk_N)$ and $sk = (vk, T)$.
2. For $i \in [N]$, parse $vk_i = (A_i, (A_{j,b}^{(i)})_{(j,b)\in[\ell\times\{0,1\}}, y_i)$.
3. Let $y = y_i$, where $i \in [N]$ is the index for which vk_i is lexicographically first.
4. If $vk \notin R$, output $\perp$ and halt.
5. Define $i^* \in [N]$ be such that $vk_{i^*} = vk$.
6. Using the trapdoor T_A for A_{i^*}, we can sample $(x_j^{(i)})_{i\in[N],j\in\{0\}\cup[\ell]}$ such that

$$\sum_{i\in[N]} A_i x_0^{(i)} + \sum_{\substack{i\in[N]\\ j\in[\ell]}} A_{j,m_j}^{(i)} x_j^{(i)} = y. \tag{4.26}$$

   That is, for $(i,j) \in [N] \times \{0\} \cup [\ell]$ other than the pair $(i^*, 0)$, we invoke algorithm $\mathsf{SampleDist}$ to sample $x_j^{(i)} \in$ independently from the discrete Gaussian distribution $\mathcal{X}$. Finally, we invoke algorithm $\mathsf{SampleCond}$ use the trapdoor T for A_{i^*} to sample $x_0^{(i^*)}$ from a distribution statistically close to the distribution $\mathcal{X}$ conditioned on Equation 4.26 being satisfied.

7. Output $\sigma = (x_j^{(i)})_{i\in[N],j\in\{0\}\cup[\ell]}$.

The verification procedure simply checks that each vector in the signature has short entries and that Equation 4.26 is satisfied.

BK-RS.Verify(R, σ, m)

1. Parse $R = (vk_1, \ldots, vk_N)$.
2. For $i \in [N]$, parse $vk_i = (A_i, (A_{j,b}^{(i)})_{(j,b)\in[\ell\times\{0,1\}}, y_i)$.
3. Parse $\sigma = (x_j^{(i)})_{i\in[N],j\in\{0\}\cup[\ell]}$.
4. For each $x_j^{(i)}$ for $i \in [N], j \in \{0\} \cup [\ell]$, if $x_j^{(i)} \notin X$ 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 4.26 above is satisfied, and otherwise reject.

Thus far we have simply described the basic ring signature scheme of [BK10]. We augment this scheme by providing an ExtractRandomness algorithm. In order to do so, we must produce "explaining randomness" that maps to the desired output vector under the algorithms SampleDist and SampleCond. We do this using algorithms ExplainDist and ExplainCond, as described in Appendix 4.10.

BK-RS.ExtractRandomness(R, sk, σ, m)

1. Parse $R = (vk_1, \ldots, vk_N)$ and $sk = (vk, T)$.
2. For $i \in [N]$, parse $vk_i = (A_i, (A_{j,b}^{(i)})_{(j,b)\in[\ell\times\{0,1\}}, y_i)$.
3. Parse $\sigma = (x_j^{(i)})_{i\in[N],j\in\{0\}\cup[\ell]}$.
4. If $vk \notin R$, output $\perp$ 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 ExplainDist to sample random coins $\rho_j^{(i)}$ that produce output $x_j^{(i)}$ under the discrete Gaussian sampling algorithm.
7. Invoke algorithm 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 4.6.2. *Under the* $\mathsf{SIS}_{q,m',\beta}$ *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.*

Proof (sketch). Completeness, anonymity and unforgeability are proven in [BK10]. It remains to show that the scheme is unclaimable. Consider the experiment described in Definition 4.3.12. The components of the signature corresponding to matrices with no trapdoor are distributed identically on the two sides of the experiment. By the correctness of algorithm ExplainDist, the components of ρ_1 and ρ_2 corresponding to identities other than vk_1, vk_2 are distributed statistically close, jointly with S and the corresponding components of the signature. It remains to consider the portions of the signature corresponding to identities vk_1 and vk_2. But Lemma 4.6.1 implies that the distribution of vectors $(x_0^{(1)}, x_0^{(2)})$ is statistically close, regardless of which trapdoor was used to sample. By the correctness of algorithm ExplainCond, the corresponding components of ρ_1 and ρ_2 are also statistically close, even conditioned on the other values in the experiment. The conclusion follows. $\square$

4.6.3 Unclaimability for the full ring signature scheme of [BK10]

The ring signature scheme 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. To achieve full unforgeability, [BK10] proceed through a sequence of four reductions to construct schemes satsifying successively stronger notions of unforgeability. We now provide a brief overview of these reductions and describe how to modify the ExtractRandomness algorithm for each scheme.

The first modified scheme appends a description of the ring to the message to be signed. This only affects that message, so the 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 Gen and is included as part of the verification key vk. During the Sign algorithm, randomness r is sampled from some distribution,[21] 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 $\mathsf{ExtractRandomness}$ 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 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 $\mathsf{ExtractRandomness}$ algorithm for the previous scheme on each of these $|m|$ messages. The final modification produces a random pad α 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 [BK10]. As above, this only affects the message to be signed, and so the $\mathsf{ExtractRandomness}$ algorithm is simply invoked on a different message.

Given the $\mathsf{ExtractRandomness}$ algorithm for the weakly-unforgeable ring signature scheme in the previous section, the modifications we have just described yield a $\mathsf{ExtractRandomness}$ algorithm for the fully-unforgeable ring signature scheme of [BK10]. It is not difficult to see that this scheme satisfies Definition 4.3.12 and is an unclaimable ring signature scheme. Consequently, we obtain the following theorem.

Theorem 4.6.3. *Under the* $\mathsf{SIS}_{q,m',\beta}$ *assumption, the ring signature scheme of [BK10] combined with the* $\mathsf{ExtractRandomness}$ *algorithm described above is an unclaimable ring signature scheme.*

[21]The distribution over which r is generated is uniform over the set of vectors with bounded ℓ_2 norm, i.e., $\{r \in \mathbb{Z}_q^{m'} : 0 < \|r\|_2 \leq \beta\}$ for some m', β.

4.7 Definitions for repudiable-and-claimable ring signatures

Definition 4.7.1 (Repudiable-and-claimable ring signature). *A* repudiable-and-claimable ring signature scheme *is a ring signature scheme with an additional quadruple of algorithms*

$$(\mathsf{Repudiate}, \mathsf{VerRepud}, \mathsf{Claim}, \mathsf{VerClaim}) \ ,$$

satisfying the five properties of correctness *(Definition 4.2.2),* repudiability *(Definition 4.7.2)* claimability *(Definition 4.7.3),* anonymity *(Definition 4.7.4), and* unforgeability *(Definition 4.7.5).*

The syntax of $(\mathsf{Repudiate}, \mathsf{VerRepud})$ *and* $(\mathsf{Claim}, \mathsf{VerClaim})$ *are as defined in Definitions 4.3.3 and 4.3.9 respectively.*

Definition 4.7.2 (Repudiability of repudiable-and-claimable ring signatures). *A ring signature scheme* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies* repudiability *if equipped with algorithms*

$$(\mathsf{Repudiate}, \mathsf{VerRepud}, \mathsf{Claim}, \mathsf{VerClaim})$$

such that for any (possibly adversarial) PPT signing algorithm $\mathcal{A}_{\mathsf{Sign}}$*, there exists a negligible function* ε *such that* (4.3) *and* (4.4) *(from Definition 4.3.3) are satisfied when* $\mathcal{O} = \{\mathsf{OSign}, \mathsf{ORpd}, \mathsf{OClaim}\}$.

Definition 4.7.3 (Claimability of repudiable-and-claimable ring signatures). *A ring signature scheme* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies* claimability *if equipped with algorithms*

$$(\mathsf{Repudiate}, \mathsf{VerRepud}, \mathsf{Claim}, \mathsf{VerClaim})$$

such that conditions 1, 2, and 3 of Definition 4.3.9 hold when $\mathcal{O} = \{\mathsf{OSign}, \mathsf{ORpd}\}$.

Definition 4.7.4 (Anonymity of repudiable-and-claimable ring signatures). *A repudiable-*

and-claimable ring signature scheme

$$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Repudiate}, \mathsf{VerRepud}, \mathsf{Claim}, \mathsf{VerClaim}))$$

satisfies anonymity against

$$\{\textit{adversarially chosen keys, attribution attacks, full key exposure}\}$$

if $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $(\{\mathsf{OSign}, \mathsf{ORpd}, \mathsf{OClaim}\}, \varnothing, \alpha)$*-anonymous (Definition 4.2.3) for, respectively,*

$$\alpha \in \{2, 1, 0\} \ .$$

Moreover, the claimable ring signature satisfies the adaptive *variants of the above anonymity definitions if* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $(\{\mathsf{OSign}, \mathsf{ORpd}, \mathsf{OClaim}\}, \{\mathsf{OSign}, \mathsf{ORpd}, \mathsf{OClaim}\}, \alpha)$*-anonymous for* $\alpha \in \{2, 1, 0\}$ *respectively.*

Definition 4.7.5 (Unforgeability of repudiable-and-claimable ring signatures). *A repudiable-and-claimable ring signature scheme*

$$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, (\mathsf{Repudiate}, \mathsf{VerRepud}, \mathsf{Claim}, \mathsf{VerClaim}))$$

is unforgeable *if* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $\{\mathsf{ORpd}, \mathsf{OClaim}\}$*-unforgeable (Definition 4.2.7).*

4.8 Completing the proof of Lemma 4.4.7

Proof (continued). It remains to show that R-RS satisfies (4.4). Suppose not, for the sake of contradiction. Then there is a PPT algorithm $\mathcal{A}_{\mathsf{S\&R}}$ such that the following probability is non-negligible:

$$\Pr\left[\begin{array}{l}(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k)\\ (\sigma, R', m, \{\xi_{vk}\}_{vk \in R' \setminus R}) \leftarrow \mathcal{A}^{\mathcal{O}}_{\mathsf{S\&R}}(R)\\ \forall vk \in R' \setminus R,\ b_{vk} \leftarrow \mathsf{VerRepud}(R', vk, \sigma, \xi_{vk})\\ b' \leftarrow \mathsf{Verify}(R', \sigma, m)\end{array} : \begin{array}{c} R' \cap R \neq \varnothing \wedge \bigwedge_{vk \in R' \setminus R} b_{vk} = 1\\ \wedge b' = 1 \wedge Q \cap \{(\cdot, m, R')\} = \varnothing\end{array}\right] \tag{4.27}$$

where $R = \{vk_1, \ldots, vk_N\}$, $\mathcal{O} = \{\mathsf{OSign}, \mathsf{ORpd}\}$, and Q is the set of queries made to oracle OSign.

Based on $\mathcal{A}_{\mathsf{S\&R}}$, we construct another adversary $\mathcal{A}'$ to Parallel VRF Game for $2N$ keys, as follows. $\mathcal{A}'$ first samples

$$(vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k)$$

and parses each vk_i as

$$vk_i = (\vec{vk}^{\,i}_{\mathsf{VRF}} = (vk^{i,1}_{\mathsf{VRF}}, vk^{i,2}_{\mathsf{VRF}}, vk^{i,3}_{\mathsf{VRF}}, vk^{i,4}_{\mathsf{VRF}}), \rho_i, \vec{\alpha}_i)\ .$$

Next, $\mathcal{A}'$ obtains $2N$ verification keys from the VRF challenger. For notational convenience in the rest of the proof, let these $2N$ keys be denoted by $\{vk^{i,3,*}_{\mathsf{VRF}}, vk^{i,4,*}_{\mathsf{VRF}}\}_{i \in [N]}$. Let the corresponding $2N$ secret keys be denoted by $\{sk^{i,3,*}_{\mathsf{VRF}}, sk^{i,4,*}_{\mathsf{VRF}}\}_{i \in [N]}$.[22]

Define vk^*_i as

$$vk^*_i = (\vec{vk}^{\,i}_{\mathsf{VRF}} = (vk^{i,1}_{\mathsf{VRF}}, vk^{i,2}_{\mathsf{VRF}}, vk^{i,3,*}_{\mathsf{VRF}}, vk^{i,4,*}_{\mathsf{VRF}}), \rho_i, \vec{\alpha}_i)\ .$$

That is, vk^*_i is identical to vk_i except that $vk^{i,3}_{\mathsf{VRF}}$ and $vk^{i,4}_{\mathsf{VRF}}$ are replaced by $vk^{i,3,*}_{\mathsf{VRF}}, vk^{i,4,*}_{\mathsf{VRF}}$.

Let $R^* = \{vk^*_1, \ldots, vk^*_N\}$. $\mathcal{A}'$ then runs $\mathcal{A}_{\mathsf{S\&R}}$ on input R^*, answering its oracle queries as follows.

- On query (i'', m'', R'') to OSign: $\mathcal{A}'$ runs the honest signing algorithm $\mathsf{R\text{-}RS.Sign}$ on

[22]Note that $\{sk^{i,3,*}_{\mathsf{VRF}}, sk^{i,4,*}_{\mathsf{VRF}}\}_{i \in [N]}$ are generated by the VRF challenger and not accessible by $\mathcal{A}'$.

input $(R'' \cup \{vk^*_{i''}\}, sk_{i''}, m'')$, with the following modification: in step 6, instead of using $sk_{i''}$ to generate y_3, τ_3, and y_4, τ_4, $\mathcal{A}'$ invokes the corresponding VRF oracle for keys $vk^{i'',3,*}_{\mathsf{VRF}}, vk^{i'',4,*}_{\mathsf{VRF}}$.

- On query (i'', σ'', R'') to ORpd: $\mathcal{A}'$ runs the honest repudiation algorithm $\mathsf{R\text{-}RS.Repudiate}$ on input $(R'' \cup \{vk^*_{i''}\}, sk_{i''}, \sigma'')$. (As noted above, sk^3_{VRF} and sk^4_{VRF} are not used in algorithm $\mathsf{R\text{-}RS.Repudiate}$, so $\mathcal{A}'$ does not need to invoke the VRF oracle here.)

Let (σ, ξ, m, R') be the output of $\mathcal{A}_{\mathsf{S\&R}}$. $\mathcal{A}'$ parses $\sigma = (\vec{\pi}, \vec{y}, \varphi)$ and $\vec{y} = (y_1, \ldots, y_4)$, then submits (R', m, φ) 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 $y_{4,i} = y_4$, $\mathcal{A}'$ outputs 0. Otherwise, $\mathcal{A}'$ outputs a random bit. Let us now consider the behavior of $\mathcal{A}'$ in the two cases where the VRF challenger's bit b is equal to 0 and equal to 1.

Case $b = 0$. In this case, the view of $\mathcal{A}_{\mathsf{S\&R}}$ is identical to the view in (4.27), so by assumption $\mathcal{A}_{\mathsf{S\&R}}$ wins the game described in (4.27) with non-negligible probability. Note that whenever $\mathcal{A}_{\mathsf{S\&R}}$ wins the game, the condition $Q \cap \{(\cdot, m, R')\} \neq \varnothing$ in (4.27) implies that $\mathcal{A}$ has not previously made an oracle query on the VRF challenge message (R', m, φ) during the query phase. Let us suppose that $\mathcal{A}_{\mathsf{S\&R}}$ wins the game described in (4.27) and consider the implications.

By definition, if $\mathsf{R\text{-}RS.Verify}$ accepts (with non-negligible probability) on input (R', σ, m), then

$$\forall i \in [|R'|],\ \mathsf{ZAP.Verify}_L(\rho_i, \pi_i, (R', m, \varphi, \vec{y})) = 1\ . \tag{4.28}$$

$R' \cap R \neq \varnothing$ from our assumption that $\mathcal{A}_{\mathsf{S\&R}}$ wins the game described in (4.27), 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 (4.28) implies that $(R', m, \varphi, \vec{y}) \in L$. Moreover, by the definition of L,

$$(R', m, \varphi, \vec{y}) \in L \Rightarrow (b_3 \vee b_4), \tag{4.29}$$

where b_3, b_4 are as defined in Definition 4.4.4. Expanding the definitions of b_3, b_4, we have that the right-hand side of (4.29) implies:

$$\begin{gathered}\exists \eta \in \{3,4\}, i^* \in [|R'|], \tau_1, \dots, \tau_4, \gamma \quad \text{s.t.} \quad \forall i' \in [|R'|], j' \in [M], \\ \mathsf{VRF.Verify}(vk_{\mathsf{VRF}}^{i^*,\eta,*}, (R', m, \varphi), y_\eta, \tau_\eta; \alpha_{j'}^{i'} \oplus \gamma) = 1 \ .\end{gathered} \tag{4.30}$$

Then applying Corollary 4.4.6 (again setting the algorithm $\mathcal{V}$ to be $\mathsf{VRF.Verify}$): (4.30) implies *either*

$$\begin{gathered}\exists \eta \in \{3,4\} \text{ and } \tau \text{ s.t.} \\ \Pr\left[\mathsf{VRF.Verify}(vk_{\mathsf{VRF}}^{i^*,\eta,*}, (R', m, \varphi), y_\eta, \tau) = 1\right] \text{ is overwhelming,}\end{gathered} \tag{4.31}$$

or a negligible probability event occurred. By the complete and unique provability of the VRF, (4.31) implies that

$$y_3 = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{i,3,*}, (R', m, \varphi)) \text{ or } y_4 = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{i,4,*}, (R', m, \varphi)) \ . \tag{4.32}$$

Chaining together the implications, we obtain that (4.32) holds with all but negligible probability conditonioned on the non-negligible-probability event of $\mathcal{A}_{\mathsf{S\&R}}$ winning the game described in (4.27).

Finally, by definition of Parallel VRF Game, when $b = 0$, for all $i \in [N]$,

$$y_{3,i} = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{i,3,*}, (R', m, \varphi)) \text{ and } y_{4,i} = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{i,4,*}, (R', m, \varphi)) \ .$$

It follows that conditioned on $b = 0$, there is a non-negligible probability that

$$\exists i^* \in [N] \text{ s.t. } y_3 = y_{3,i^*} \text{ or } y_4 = y_{4,i^*} \ . \tag{4.33}$$

Recall that (4.33) is the trigger condition for $\mathcal{A}'$ to output 0. Therefore, when $b = 0$, $\mathcal{A}'$ 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 $\mathcal{A}$ outputs a random bit.

Thus, $\mathcal{A}'$ wins Parallel VRF Game with non-negligible probability. This contradicts the security of the VRF. We therefore conclude that $\mathsf{R\text{-}RS}$ satisfies (4.4). The lemma follows. $\square$

4.9 Deferred proofs: anonymity of R-RS

Lemma 4.4.9. $\mathsf{R\text{-}RS}$ is satisfies adaptive anonymity against adversarially chosen keys (Definition 4.3.4).

Proof. Suppose that this is not the case. Then there exists some $N = \mathsf{poly}(k)$ and PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that the following probability is non-negligibly greater than 1/2, where I is the set of queries to the corruption oracle and $\mathcal{O} = \{\mathsf{OSign}, \mathsf{ORpd}\}$:

$$\Pr\left[\begin{array}{l} (vk_1, sk_1), \ldots, (vk_N, sk_N) \leftarrow \mathsf{Gen}(1^k) \\ ((m^*, i_0^*, i_1^*, R^*), \mathfrak{s}) \leftarrow \mathcal{A}_1^{\mathcal{O},\mathsf{Corr}}(vk_1, \ldots, vk_N) \\ b \leftarrow \{0,1\} \\ \sigma \leftarrow \mathsf{Sign}(R^* \cup \{vk_{i_0^*}, vk_{i_1^*}\}, sk_{i_b^*}, m^*) \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O},\mathsf{Corr}}(\mathfrak{s}, \sigma) \end{array} : b' = b \wedge \{i_0^*, i_1^*\} \cap I = \varnothing \right].$$

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 $sk_{i_1^*} = ((sk_{\mathsf{VRF}}^{i_1^*,1}, \ldots, sk_{\mathsf{VRF}}^{i_1^*,4}), vk_{i_1^*})$, we set*

$$y_2 = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{i_1^*,2}, (R, m, \varphi))$$

and

$$y_4 = \mathsf{VRF.Eval}(sk_{\mathsf{VRF}}^{i_1^*,4}, (R, m, \varphi)).$$

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 σ, the ZAPs in the* Sign *procedure are proven with respect to the witnesses for party i_1^* rather than i_0^*. That is, parsing $sk_{i_1^*} = ((sk_{\mathsf{VRF}}^{i_1^*,1}, \ldots, sk_{\mathsf{VRF}}^{i_1^*,4}), vk_{i_1^*})$, in the invocation of* Sign *to generate σ, let $\tau_2' = \mathsf{VRF.Prove}(sk_{\mathsf{VRF}}^{i_1^*,2}, (R, m, \varphi))$ and $\tau_4' = \mathsf{VRF.Prove}(sk_{\mathsf{VRF}}^{i_1^*,4}, (R, m, \varphi))$. For each $i \in [N]$, in step 7 of* Sign*, generate π_i by invoking*

$$\mathsf{ZAP.Prove}(\rho_i, \mathsf{stmt}, (i_1^*, \perp, \tau_2', \perp, \tau_4', \gamma))$$

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 $\sigma = (\vec{\pi}, (y_1, \ldots, y_4))$, 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 $\sigma = (\vec{\pi}, (y_1, \ldots, y_4))$, 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. □

4.10 Explaining randomness of discrete Gaussian samples

Our construction requires "explaining" algorithms ExplainDist and ExplainCond. ExplainDist must, on input (x, y), output ρ distributed according to the conditional distribution

$$\left\{\rho \mid \mathsf{SampleDist}(x;\rho) = y\right\} .$$

Similarly, ExplainCond must, on input (x, y), output ρ distributed according to

$$\left\{\rho \mid \mathsf{SampleCond}(x;\rho) = y\right\} .$$

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 [BK10]:

- SampleDist is instantiated by the function $\mathsf{SampleD}(B, s, c)$ defined in [GPV07, §4.2].[23]
- SampleCond is instantiated by the function $\mathsf{SampleISIS}(A, T, s, u)$ defined in [GPV07, §5.3.2].

We first recall the basis randomization technique of [CHKP10]. In the following, tildes denote Gram-Schmidt orthogonalization.

Lemma 4.10.1 (Lemma 3.3 in [CHKP10]). *There exists a probabilistic polynomial-time algorithm* $\mathsf{RandBasis}(T, s)$ *that takes as input a basis T of an m'-dimensional integer lattice and a parameter $s \geq \|\tilde{T}\| \cdot \omega(\sqrt{\log n})$, and outputs a basis T' for the same lattice, such that:*

[23] We cite the full version here for the detailed description of the function; the conference version is [GPV08].

1. *With overwhelming probability,* $\|\mathsf{RandBasis}(T, s)\| \leq s \cdot \sqrt{m'}$

2. *For any pair of bases* T_1, T_2 *for the same lattice and any* $s \geq \max(\|\tilde{T_1}\|, \|\tilde{T_2}\|) \cdot \omega(\sqrt{\log n})$*, the outputs of* $\mathsf{RandBasis}(T_1, s)$ *and* $\mathsf{RandBasis}(T_2, s)$ *have negligible statistical distance.*

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 Sample$\mathbb{Z}$ which is also defined in [GPV07]. We describe these three algorithms in the order they are listed in the preceding sentence.

Definition 4.10.1. 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. *Let* $T' \leftarrow \mathsf{RandBasis}(T, s)$.

2. *By Gaussian elimination, choose an arbitrary* t *such that* $At = u \pmod q$.

3. *Sample* $v \leftarrow \mathsf{SampleD}(T', s, -t)$.

4. *Output* $e = t + v$.

Claim 4.10.2. *There is an efficient algorithm for* ExplainCond *if there is an efficient algorithm for* ExplainDist.

Proof. Essentially, the claim follows from the fact that Step 3 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 2) 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 $e - t$. □

Definition 4.10.2. 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.*

1. *Let $v_n := 0$ (the zero vector) and $c_n := c$. For $i = n, \dots, 1$:*

    (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 $z_i \leftarrow \mathsf{Sample}\mathbb{Z}(s'_i, c'_i)$.*

    (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 4.10.3. *There is an efficient algorithm for* ExplainDist *if there is an efficient algorithm to "explain* Sample$\mathbb{Z}$*" — i.e., an efficient algorithm that, on input (s, c, x), samples from the following distribution:*

$$\left\{\rho \mid \mathsf{Sample}\mathbb{Z}(s, c; \rho) = x\right\} . \tag{4.34}$$

Proof (sketch). Each output of SampleDist on a given input induces a unique set of z_is — in other words, as noted in [GPV07], there is a bijective correspondence between the random choices of the z_is and the output. After recovering these z_is, it remains only to explain the randomness of Sample$\mathbb{Z}$ conditioned on inputs (s'_i, c'_i) and outputs z_i for each i. □

Definition 4.10.3. Sample$\mathbb{Z}$ *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, $t(n) = \log(n)$;* Ber(p) *denotes the Bernoulli distribution with probability p of outputting 1; and $\rho_s(\cdot)$ 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 $b \leftarrow \mathsf{Ber}(\rho_s(x - c))$.*

3. *If $b = 1$, output x. Else, go back to step 1.*

Claim 4.10.4. *There is an efficient algorithm that, on input (s, c, x), samples from the distribution described in* (4.34).

Proof (sketch). Sample$\mathbb{Z}$ consists of rejection sampling based on a biased coinflip, where the bias depends on the present sample x. The randomness ρ to explain a particular output of

$\mathsf{Sample}\mathbb{Z}$ can be thought to consist of a series of "rejected" samples $x_1, \ldots, x_{\ell-1}$ followed by the "accepted" sample x_ℓ (which must be equal to x). The length ℓ of this sequence follows a geometric distribution parametrized by the *expected* bias β over the entire domain X. That is,

$$\beta = \frac{1}{|X|} \sum_{x \in X} \rho_s(x - c) \ .$$

Once a value of ℓ 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 $b \leftarrow \mathsf{Ber}(\rho_s(x' - c))$.
3. If $b = 0$, then set $x_i := x'$ and continue. Else, go back to Step 1.

$\square$

Part III

Differential privacy

Chapter 5

Private shortest paths and distances

This chapter is based on the work [Sea16].

5.1 Introduction

5.1.1 Differential privacy and our model

Privacy-preserving data analysis is concerned with releasing useful aggregate information from a database while protecting sensitive information about individuals in the database. *Differential privacy* [DMNS06] provides strong privacy guarantees while allowing many types of queries to be answered approximately. Differential privacy requires that for any pair of *neighboring* databases x and y, the output distributions of the mechanism on database x and database y are very close. In the traditional setting, databases are collections of data records and are considered to be neighboring if they are identical except for a single record, which is thought of as the information associated with a single individual. This definition guarantees that an adversary can learn very little about any individual in the database, no matter what auxiliary information the adversary may possess.

A new model We introduce a model of differential privacy for the setting in which databases are weighted graphs $(\mathcal{G}, w)$. In our setting, the graph topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is assumed to be public, and only the edge weights $w : \mathcal{E} \to \mathbb{R}^+$ must be kept private. Individuals are assumed to have only bounded influence on the edge weights. Consequently, two weight functions on the same unweighted graph are considered to be neighbors if they have ℓ_1 distance at most one.

The model is well suited to capture the setting of a navigation system which has access to current traffic data and uses it to direct drivers. The travel times along routes provided by the system should be as short as possible. However, the traffic data used by the system may consist of private information. For instance, navigation tools such as Google Maps and Waze estimate traffic conditions based on GPS locations of cars or based on traffic reports by users. On the other hand, the topology of the road map used by the system is non-private, since it is a static graph readily available to all users. We would like to provide information about paths and distances in the network without revealing sensitive information about the edge weights. In this paper we introduce a variant of differential privacy suitable for studying network routing, and provide both algorithms and lower bounds.

5.1.2 Our results

We will consider two classic problems in this model which are relevant for routing. First, we are interested in finding *short paths* between specified pairs of vertices. We cannot hope always to release the shortest path while preserving privacy, but we would like to release a path that is not much longer than the shortest path. Here the *approximation error* is the difference in length between the released path and the shortest path. Second, we would like to release *distances* between pairs of vertices. The *approximation error* here is the absolute difference between the released distance and the actual distance. As we discuss below, releasing an accurate distance estimate for a single pair of vertices in our model is a straightforward application of the Laplace mechanism of [DMNS06]. We focus on the more difficult problem of releasing *all-pairs distances* privately with low error.

Interestingly, all of our error bounds are independent of the sum of the edge weights $\|w\|_1$, which corresponds most naturally to the size of the database under the usual setting for differential privacy. Instead, they depend only on the number of vertices $|\mathcal{V}|$ and edges $|\mathcal{E}|$ in the graph and the privacy parameters ε and δ. Our error bounds constitute additive error. Consequently, if the edge weights are large, the error will be small in comparison.

Approximate shortest paths We consider the problem of privately releasing an approximate *shortest path*. We provide a strong reconstruction based lower bound, showing that in general it is not possible under differential privacy to release a short path between a pair of vertices with additive error better than $\Omega(|\mathcal{V}|)$. Our lower bound is obtained by reducing the problem of reconstructing many of the rows of a database to the problem of finding a path with low error.

We also show that a simple ε-differentially private algorithm based on the Laplace mechanism of [DMNS06] comes close to meeting this bound. If the minimum-weight path between a pair of vertices s, t consists of at most k edges, then the weight of the path released by our algorithm is greater by at most $O(k \log |\mathcal{V}|)/\varepsilon$. Since $k < |\mathcal{V}|$, the weight of the released path is $O(|\mathcal{V}| \log |\mathcal{V}|)/\varepsilon$ greater than optimal. Note that when the edge weights are large, the length of a path can be much larger than $(|\mathcal{V}| \log |\mathcal{V}|)/\varepsilon$, in which case our algorithm provides a good approximation. Moreover, for many networks we would expect that shortest paths should be compact and consist of few vertices, in which case the accuracy obtained will be much greater. The algorithm releases not only a single path but short paths between every pair of vertices without loss of additional privacy or accuracy.

Approximate all pairs distances For the problem of privately releasing all-pairs *distances*, standard techniques yield error $O(|\mathcal{V}| \log |\mathcal{V}|)/\varepsilon$ for each query under ε-differential privacy. We obtain improved algorithms for two special classes of graphs, trees and arbitrary graphs with edges of bounded weight.

For trees we show that a simple recursive algorithm can release all-pairs distances with error $O(\log^{2.5} |\mathcal{V}|)/\varepsilon$. Implicitly, the algorithm finds a collection $\mathcal{C}$ of paths with two properties. Every edge is contained in at most $\log |\mathcal{V}|$ paths, and the unique path between any pair of vertices can be formed from at most $4 \log |\mathcal{V}|$ paths in $\mathcal{C}$ using the operations of concatenation and subtraction. The first property implies that the query consisting of the lengths of all of the paths in $\mathcal{C}$ has global ℓ_1 sensitivity $\log|\mathcal{V}|$, so we can release estimates of the lengths of all paths in $\mathcal{C}$ with error roughly $\log|\mathcal{V}|$ using the standard Laplace mechanism [DMNS06]. The second property implies that we can use these estimates to compute the distance between any pair of vertices without too much increase in error. We first reduce the approximate all-pairs distances problem to the problem of approximating all distances from a single fixed vertex. We then solve the single-source problem recursively, repeatedly decomposing the tree into subtrees of at most half the size.

In the special case of the path graph, releasing approximate all-pairs distances is equivalent to query release of threshold functions on the domain $X = \mathcal{E}$. The results of [DNPR10] yield the same error bound that we obtain for computing distances on the path graph. Consequently our algorithm for all-pairs distances on trees can be viewed as a generalization of a result for private query release of threshold functions.

For bounded-weight graphs we show that we can generate a set of vertices $\mathcal{Z} \subset \mathcal{V}$ such that distances between all pairs of vertices in $\mathcal{Z}$ suffice to estimate distances between all pairs of vertices in $\mathcal{V}$. The required property on $\mathcal{Z}$ is that every vertex $v \in \mathcal{V}$ is connected to a vertex $z_v \in \mathcal{Z}$ by a path consisting of few vertices. Since we can always find a set $\mathcal{Z}$ that is not too large, we can release distances between all pairs of vertices in $\mathcal{Z}$ with relatively small error. The distance between a pair of vertices $u, v \in \mathcal{V}$ can then be approximated by the distance between nearby vertices $z_u, z_v \in \mathcal{Z}$. In general, if edge weights are in $[0, M]$ for $\frac{1}{|\mathcal{V}|} < M\varepsilon < |\mathcal{V}|$, then we can release all pairs shortest paths with an additive error of $\tilde{O}(\sqrt{|\mathcal{V}|M\varepsilon^{-1}\log(1/\delta)})$ per path under (ε, δ)-differential privacy for any $\varepsilon, \delta > 0$. Note that the distance between a pair of vertices can be as large as $|\mathcal{V}| \cdot M$, so this error bound is nontrivial. For specific graphs we can obtain better bounds by finding a smaller set $\mathcal{Z}$ satisfying the necessary requirements.

Additional problems We also consider two other problems in this model, the problem of releasing a nearly *minimal spanning tree* and the problem of releasing an almost *minimum weight perfect matching*. For these problems, the *approximation error* is the absolute difference in weight between the released spanning tree or matching and the minimum-weight spanning tree or matching.

Using a similar argument to the shortest path results of Section 5.5, we provide lower bounds and algorithms for these problems. Through a reduction from the problem of reconstructing many rows in a database, we show that it is not possible under differential privacy to release a spanning tree or matching with error better than $\Omega(|\mathcal{V}|)$. Using a simple algorithm based on the Laplace mechanism of [DMNS06], we can privately release a spanning tree or matching with error roughly $(|\mathcal{V}| \log |\mathcal{V}|)/\varepsilon$. These results are presented in Appendix 5.7.

Future directions In this work we describe differentially private algorithms and lower bounds for several natural graph problems related to network routing. We would like to explore how well our algorithms perform in practice on actual road networks and traffic data. We would also like to develop new algorithms for additional graph problems and to design improved all-pairs distance algorithms for additional classes of networks. In addition, lower bounds for accuracy of private all-pairs distances would be very interesting.

Scaling Our model requires that we preserve the indistinguishability of two edge weight functions which differ by at most 1 in ℓ_1 norm. The constant 1 here is arbitrary, and the error bounds in this paper scale according to it. For instance, if a single individual can only influence edge weights by $1/|\mathcal{V}|$ rather than 1 in ℓ_1 norm, then we can privately find a path between any pair of vertices whose length is only $O(\log |\mathcal{V}|)/\varepsilon$ longer than optimal rather than $O(|\mathcal{V}| \log |\mathcal{V}|)/\varepsilon$. The other results in this paper can be scaled similarly.

5.1.3 Previous work on graphs and differential privacy

Why a new model? The two main models which have been considered for applying differential privacy to network structured data are *edge* and *node* differential privacy [HLMJ09, BBDS13, KNRS13, KRSY14]. Under edge differential privacy, two graphs are considered to be neighbors if one graph can be obtained from the other by adding or removing a single edge. With node differential privacy, two graphs are neighbors if they differ only on the set of edges incident to a single vertex.

However, the notions of edge and node differential privacy are not well suited to shortest path and distance queries. Consider an algorithm that releases a short path between a specified pair of vertices. Any useful program solving this problem must usually at least output a path of edges which are in the graph. But doing so blatantly violates privacy under both the edge and node notions of differential privacy, since the released edges are private information. Indeed, an algorithm cannot release subgraphs of the input graph without violating both edge and node differential privacy because this distinguishes the input from a neighboring graph in which one of the released edges is removed.

What if we simply want to release the distance between a pair of vertices? In general it is not possible to release approximate distances with meaningful utility under edge or node differential privacy, since changing a single edge can greatly change the distances in the graph. Consider the unweighted path graph $\mathcal{P}$ and any pair of adjacent vertices x, y on the path. Removing edge (x, y) disconnects the graph, increasing the distance between x and y from 1 to ∞. Even if the graph remains connected, the removal of an edge does not preserve approximate distances. Consider any pair of adjacent vertices x, y on the cycle graph $\mathcal{C}$. Here, removing edge (x, y) increases the distance between x and y from 1, the smallest possible distance, to $|\mathcal{V}| - 1$, the largest possible distance. Hence, the edge and node notions of differential privacy do not enable the release of approximate distances. This inadequacy motivates the new notion of differential privacy for graphs introduced in this work.

A histogram formulation A database consisting of elements from a data universe $\mathcal{U}$ can be described by a vector $D \in \mathbb{N}^{|\mathcal{U}|}$, where the ith component of the vector corresponds to the number of copies of the ith element of $\mathcal{U}$ in the database. More generally, we can allow the database to be a point in $\mathbb{R}^{|\mathcal{U}|}$, which corresponds to allowing a fractional number of occurrences of each element. Since an edge weight function w is an element of $\mathbb{R}^{|\mathcal{E}|}$ and the notions of sensitivity coincide, we can rephrase our model in the standard differential privacy framework.

Consequently, we can use existing tools for privately answering low sensitivity queries to yield incomparable results to those presented in this paper for the problem of releasing all-pairs distances with low error. If all edge weights are integers and the sum $\|w\|_1$ of the edge weights is known, we can release all-pairs distances with additive error

$$\tilde{O}(\sqrt{\|w\|_1 \cdot \log |\mathcal{V}|} \log^{1.5}(1/\delta)/\varepsilon)$$

except with probability δ using the (ε, δ)-differentially private boosting mechanism of [DRV10]. The assumption that $\|w\|_1$ is known is not problematic, since we can privately release a good approximation. We can also extend the algorithm to noninteger edge weights for the queries in our setting, although we obtain a worse error bound. We do this by modifying the base synopsis generator of [DRV10] to produce a database in $(\tau\mathbb{N})^{|\mathcal{E}|}$ whose fractional counts are integer multiples of $\tau = \alpha/(2|\mathcal{V}|)$, where α will be the obtained additive approximation. The modified algorithm releases all-pairs distances with error $\tilde{O}(\sqrt[3]{\|w\|_1 \cdot |\mathcal{V}|} \log^{4/3}(1/\delta)/\varepsilon^{2/3})$.

This error bound has a better dependence on $|\mathcal{V}|$ than the algorithms for all-pairs distances in general and weighted graphs described in Section 5.4. However, it has a substantial dependence on $\|w\|_1$, while the error bound for all algorithms in the remainder of this paper are independent of $\|w\|_1$. In addition, the running time of the algorithm of [DRV10] is exponential in the database size, while all algorithms described below run in polynomial time.

Additional related work While the private edge weight model explored in this work is new, a few previous works have considered problems on graphs in related models. Nissim, Raskhodnikova and Smith [NRS07] consider the problem of computing the cost of the minimum spanning tree of a weighted graph. They introduce the notion of smooth sensitivity, which they use to compute the approximate MST cost. In their work, edge weights are bounded, and each weight corresponds to the data of a single individual. In contrast, we allow unbounded weights but assume a bound on the effect an individual can have on the weights.

Hsu et al. [HHR+14] consider the problem of privately finding high-weight matchings in bipartite graphs. In their model, which captures a private version of the allocation problem, two weightings of the complete bipartite graph are neighbors if they differ only in the weights assigned to the edges incident to a single left-vertex, which correspond to the preferences of a particular agent. They show that the problem is impossible to solve under the standard notion of differential privacy. They work instead under the relaxed notion of "joint differential privacy," in which knowledge of the edges of the matching which correspond to some of the left-vertices cannot reveal the weights of edges incident to any of the remaining left-vertices.

A series of works has explored the problem of privately computing the size of all cuts $(S, \overline{S})$ in a graph. Gupta, Roth and Ullman [GRU12] show how to answer cut queries with $O(|\mathcal{V}|^{1.5})$ error. Blocki et al. [BBDS12] improve the error for small cuts. Relatedly, Gupta et al. [GLM+10] show that we can privately release a cut of close to minimal size with error $O(\log|\mathcal{V}|)/\varepsilon$, and that this is optimal. Since the size of a cut is the number of edges crossing the cut, it can also be viewed as the sum of the weights of the edges crossing the cut in a $\{0, 1\}$-weighting of the complete graph. Consequently the problem can be restated naturally in our model.

As we have discussed, the problem of approximating all threshold functions on a totally ordered set is equivalent to releasing approximate distances on the path graph. In addition to the work of Dwork et al. [DNPR10] described above, additional bounds and reductions for query release and learning of threshold functions are shown in Bun et al. [BNSV15].

5.2 The privacy model

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote an undirected graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$, and let $w : \mathcal{E} \to \mathbb{R}^+$ be a *weight function.* (The shortest path results in Section 5.5 also apply to directed graphs.) Let $V = |\mathcal{V}|$ and $E = |\mathcal{E}|$ be the number of vertices and edges, respectively.

Let $\mathcal{P}_{xy}$ denote the set of paths between a pair of vertices $x, y \in \mathcal{V}$. For any path $P \in \mathcal{P}_{xy}$, the weight $w(P)$ is the sum $\sum_{e \in P} w(e)$ of the weights of the edges of P. The distance $d_w(x, y)$ from x to y denotes the weighted distance $\min_{P \in \mathcal{P}_{xy}} w(P)$. We will denote the unweighted or hop distance from x to y by $h(x, y) = \min_{P \in \mathcal{P}_{xy}} \ell(P)$, where the hop length $\ell(P)$ of path $P = (v_0, \dots, v_\ell)$ is the number ℓ of edges on the path. Let the shortest path $\mathrm{SP}_w(x, y)$ denote a path from x to y of minimum possible weight $w(\mathrm{SP}_w(x, y)) = d_w(x, y)$.

We now formally define differential privacy in the private edge weight model.

Definition 5.2.1. *For any edge set $\mathcal{E}$, two weight functions $w, w' : \mathcal{E} \to \mathbb{R}^+$ are* ***neighboring****, denoted $w \sim w'$, if*

$$\|w - w'\|_1 = \sum_{e \in \mathcal{E}} |w(e) - w'(e)| \leq 1.$$

Definition 5.2.2. *For any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\mathcal{A}$ be an algorithm that takes as input a weight function $w : \mathcal{E} \to \mathbb{R}^+$. $\mathcal{A}$ is (ε, δ)-**differentially private** on $\mathcal{G}$ if for all pairs of neighboring weight functions w, w' and for all sets of possible outputs S, we have that*

$$\Pr[\mathcal{A}(w) \in S] \leq e^{\varepsilon} \Pr[\mathcal{A}(w') \in S] + \delta.$$

If $\delta = 0$ we say that $\mathcal{A}$ is ε-differentially private on $\mathcal{G}$.

For a class $\mathcal{C}$ of graphs, we say that $\mathcal{A}$ is (ε, δ)-differentially private on $\mathcal{C}$ if $\mathcal{A}$ is (ε, δ)-differentially private on $\mathcal{G}$ for all $\mathcal{G} \in \mathcal{C}$.

We now define our accuracy criteria for the approximate shortest paths and distances problems.

Definition 5.2.3. *For the shortest path problem, the* ***error*** *of a path $P \in \mathcal{P}_{xy}$ between vertices x, y is the difference $w(P) - d_w(x, y)$ between the length of P and the length of the shortest path between x and y.*

Definition 5.2.4. *For the approximate distances problem, the* ***error*** *is the absolute difference between the released distance between a pair of vertices x, y and the actual distance*

$d_w(x, y)$.

5.3 Preliminaries

We will now introduce a few basic tools which will be used throughout the remainder of this paper. A number of differential privacy techniques incorporate noise sampled according to the Laplace distribution. We define the distribution and state a concentration bound for sums of Laplace random variables.

Definition 5.3.1. *The **Laplace distribution** with scale b, denoted $\mathsf{Lap}(b)$, is the distribution with probability density function given by*

$$p(x) = \frac{1}{2b} \exp(-|x|/b).$$

If Y is distributed according to $\mathsf{Lap}(b)$, then for any $t > 0$ we have that $\Pr[|Y| > t \cdot b] = e^{-t}$.

Lemma 5.3.1 (Concentration of Laplace random variables [CSS10]). *Let $X_1, \ldots, X_t$ be independent random variables distributed according to $\mathsf{Lap}(b)$, and let $X = \sum_i X_i$. Then for all $\gamma \in (0, 1)$ we have that with probability at least $1 - \gamma$,*

$$|X| < 4b\sqrt{t} \ln(2/\gamma) = O(b\sqrt{t} \log(1/\gamma)).$$

One of the first and most versatile differentially private algorithms is the Laplace mechanism, which releases a noisy answer with error sampled from the Laplace distribution with scale proportional to the sensitivity of the function being computed.

Definition 5.3.2. *For any function $f : \mathcal{X} \to \mathbb{R}^k$, the **sensitivity***

$$\Delta f = \max_{\substack{w, w' \in \mathcal{X} \\ w \sim w'}} \|f(w) - f(w')\|_1$$

is the largest amount f can differ in ℓ_1 norm between neighboring inputs.

In our setting we have $\mathcal{X} = (\mathbb{R}^+)^{\mathcal{E}}$.

Lemma 5.3.2 (Laplace mechanism [DMNS06]). *Given any function $f : \mathcal{X} \to \mathbb{R}^k$, the Laplace mechanism on input $w \in \mathcal{X}$ independently samples $Y_1, \ldots, Y_k$ according to $\mathsf{Lap}(\Delta f/\varepsilon)$ and outputs*

$$\mathcal{M}_{f,\varepsilon}(w) = f(w) + (Y_1, \ldots, Y_k).$$

The Laplace mechanism is ε-differentially private.

Finally, we will need the following results on the composition of differentially private algorithms.

Lemma 5.3.3 (Basic Composition, e.g. [DKM+06]). *For any ε, $\delta \geq 0$, the adaptive composition of k (ε, δ)-differentially private mechanisms is $(k\varepsilon, k\delta)$-differentially private.*

Lemma 5.3.4 (Composition [DRV10, DR13]). *For any $\varepsilon, \delta, \delta' \geq 0$, the adaptive composition of k (ε, δ)-differentially private mechanisms is $(\varepsilon', k\delta + \delta')$-differen-tially private for*

$$\varepsilon' = \sqrt{2k\ln(1/\delta')} \cdot \varepsilon + k\varepsilon(e^{\varepsilon} - 1)$$

which is $O(\sqrt{k\ln(1/\delta')} \cdot \varepsilon)$ provided $k \leq 1/\varepsilon^2$. In particular, if $\varepsilon' \in (0,1), \delta, \delta' \geq 0$, the composition of k (ε, δ)-differentially private mechanisms is $(\varepsilon', k\delta + \delta')$-differentially private for $\varepsilon = O(\varepsilon'/\sqrt{k\ln(1/\delta')})$.

5.4 Computing distances

In this section we consider the problem of releasing distance oracle queries in the private edge weights model. Since neighboring weight functions differ by at most 1 in ℓ_1 norm, the weight of any path also changes by at most 1. Consequently, a single distance oracle query is sensitivity-1, and so we can use the Laplace mechanism (Lemma 5.3.2) to answer it privately after adding noise proportional to $1/\varepsilon$. However, what if we would like to learn all-pairs distances privately?

There are V^2 pairs (s, t) of vertices, so we can achieve ε-differential privacy by adding to each query Laplace noise proportional to V^2/ε. We can do better using approximate differential privacy ($\delta > 0$) and Lemma 5.3.4 (Composition). Adding Laplace noise proportional to $1/\varepsilon'$ to each query results in a mechanism that is $(V\varepsilon'\sqrt{2\ln(1/\delta)} + V^2\varepsilon'(e^{\varepsilon'} - 1), \delta)$-differentially private for any $\delta > 0$. Taking $\varepsilon' = \varepsilon/O(V\sqrt{\ln 1/\delta})$ for $\varepsilon < 1$, we obtain a mechanism that is (ε, δ)-differentially private by adding $O(V\sqrt{\ln 1/\delta})/\varepsilon$ noise to each query.

The other natural approach is to release an ε-differentially private version of the graph by adding $\mathsf{Lap}(1/\varepsilon)$ noise to each edge. This will be the basis for our approach in Algorithm 10 for computing approximate shortest paths. With probability $1-\gamma$, all E Laplace random variables will have magnitude $(1/\varepsilon)\log(E/\gamma)$, so the length of every path in the released synthetic graph is within $(V/\varepsilon)\log(E/\gamma)$ of the length of the corresponding path in the original graph. Therefore with probability $1-\gamma$ we have that all pairs distances in the released synthetic graph will be within $(V/\varepsilon)\log(E/\gamma)$ of the corresponding distances in the original graph.

Both of these approaches result in privately releasing all pairs distances with additive error roughly V/ε. It is natural to ask whether this linear dependance on V is the best possible. In the remainder of this section we present algorithms which substantially improve on this bound for two special classes of graphs, trees and arbitrary graphs with edges of bounded weight. For trees we can obtain all-pairs distances with error $\text{polylog}(V)/\varepsilon$, and for graphs with edges in $[0, M]$ we can obtain all-pairs distances with error roughly $\sqrt{VM/\varepsilon}$.

5.4.1 Distances in trees

For trees it turns out to be possible to release all-pairs distances with far less error. We will first show a single-source version of this result for rooted trees, which we will then use to obtain the full result. The idea is to split the tree into subtrees of at most half the size of the original tree. As long as we can release the distance from the root to each subtree with small error, we can then recurse on the subtrees.

The problem of privately releasing all-pairs distances for the path graph can be restated as the problem of privately releasing threshold functions for a totally ordered data universe of size $E = V - 1$. Dwork et al. [DNPR10] showed that we can privately maintain a running counter for T timesteps, with probability $1 - \gamma$ achieving additive accuracy of $O(\log(1/\gamma)\log^{2.5} T/\varepsilon)$ at each timestep. This algorithm essentially computes all threshold queries for a totally ordered universe of size T, which is equivalent to computing the distance from an endpoint to each other vertex of the path graph. We match the accuracy of the [DNPR10] algorithm and generalize the result to arbitrary trees.

In Appendix 5.6 we provide an alternate algorithm for the path graph which achieves the same asymptotic bounds. This result can be viewed as a restatement of the [DNPR10] algorithm.

Theorem 5.4.1 (Single-source distances on rooted trees). *Let $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ be a tree with root vetex v_0, and let $\varepsilon > 0$. Then there is an algorithm that is ε-differentially private on $\mathcal{T}$ that on input edge weights $w : \mathcal{E} \to \mathbb{R}^+$ releases approximate distances from v_0 to each other vertex, where with probability $1 - \gamma$ the approximation error on each released distance is*

$$O(\log^{1.5} V \cdot \log(1/\gamma))/\varepsilon$$

for any $\gamma \in (0, 1)$.

Proof. Given the tree $\mathcal{T}$ and root v_0, we will partition $\mathcal{V}$ into subtrees of size at most $V/2$ as shown in Figure 5-1 and recursively obtain distances in each subtree. There exists some vertex v^* such that the subtree rooted at v^* contains more than $V/2$ vertices while the subtree rooted at each child of v^* has at most $V/2$ vertices. The topology of the graph is public, so we can release vertex v^*. Let $v_1, \ldots, v_t$ be the children of v^*, and let $\mathcal{T}_i = (\mathcal{V}_i, \mathcal{E}_i)$ be the subtree rooted at v_i for each $i \in [t]$. Let $\mathcal{T}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ be the subtree rooted at v_0 consisting of the remaining vertices $\mathcal{V} \setminus (\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_t)$.

We can compute and release distances between the following pairs of vertices, adding $\mathsf{Lap}(\log V/\varepsilon)$ noise to each distance:

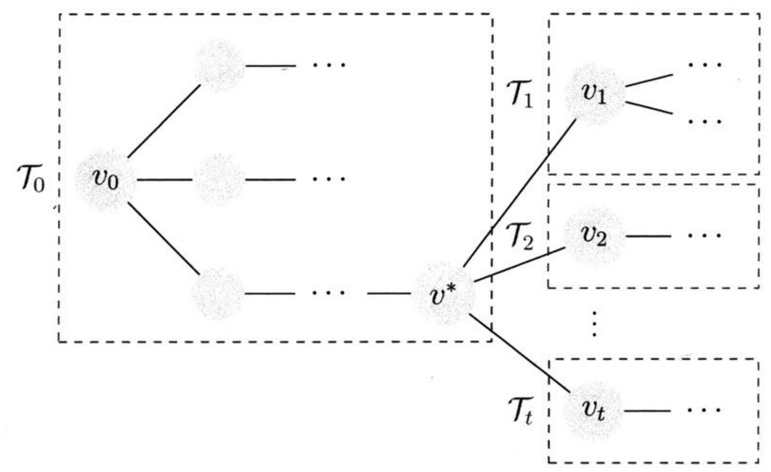


Figure 5-1: The partition used in Algorithm 8 to separate a tree into subtrees of size at most $V/2$.

- (v_0, v^*)
- (v^*, v_i) for each $i \in [t]$

Since v^* is the parent of $v_1, \ldots, v_t$ in the tree rooted at v_0, the path from v_0 to v^* in T contains none of the edges (v^*, v_i) for $i \in [t]$, so the function which releases these $t+1$ distances is sensitivity-1.

We then recursively repeat the procedure on each subtree $\mathcal{T}_0, \ldots, \mathcal{T}_t$ until we reach trees containing only a single vertex, adding $\mathsf{Lap}(\log V/\varepsilon)$ noise to each released value. Each subtree has size at most $V/2$, so the depth of recursion is bounded by $\log V$. The subtrees $\mathcal{T}_0, \ldots, \mathcal{T}_t$ are also disjoint. Consequently the function which releases all of the distances at depth d in the recursion has sensitivity 1 for any d. Therefore the function which releases all distances queried in this recursive procedure has sensitivity at most $\log V$. Since we add $\mathsf{Lap}(\log V/\varepsilon)$ noise to each coordinate of this function, the algorithm outlined above is an instantiation of the Laplace mechanism (Lemma 5.3.2) and is ε-differentially private.

We now show how these queries suffice to compute the distance from root vertex v_0 to each other vertex with small error. The algorithm samples at most $2V$ Laplace random variables distributed according to $\mathsf{Lap}(\log V/\varepsilon)$, so by a union bound, with probability $1-\gamma$ all of these have magnitude $O(\log V \log(V/\gamma))/\varepsilon$. Consequently to obtain an error bound of

Algorithm 8 Rooted tree distances

```
1: Inputs: Tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, root $v_0 \in \mathcal{V}$, edge weights $w : \mathcal{E} \to \mathbb{R}^+$, and number of vertices
   $n$ of original tree.
2: Let $v^*$ be the unique vertex such that the subtree rooted at $v^*$ has more than $V/2$ vertices
   while the subtree rooted at each child of $v^*$ has at most $V/2$ vertices.
3: Let $v_1, \dots, v_t$ be the children of $v^*$.
4: Let $\mathcal{T}_i$ be the subtree rooted at $v_i$ for $i \in [t]$, and let $\mathcal{T}_0 = \mathcal{T} \setminus \{\mathcal{T}_1, \dots, \mathcal{T}_t\}$.
5: Sample $X_{v^*,\mathcal{T}} \leftarrow \mathsf{Lap}(\log V/\varepsilon)$ and let $d(v^*, \mathcal{T}) = d_w(v_0, v^*) + X_{v^*,\mathcal{T}}$.
6: Let $d(v_0, \mathcal{T}) = 0$.
7: for each $i \in [t]$ do
8:     Sample $X_{v_i,\mathcal{T}} \leftarrow \mathsf{Lap}(\log V/\varepsilon)$
9:     Let $d(v_i, \mathcal{T}) = d(v^*, \mathcal{T}) + w((v^*, v_i)) + X_{v_i,\mathcal{T}}$.
10: end for
11: Recursively compute distances in each subtree $\mathcal{T}_0, \dots, \mathcal{T}_t$.
12: for each vertex $v \in \mathcal{V}$ do
13:     if $v \in \mathcal{T}_i$ then
14:         Let $d(v, \mathcal{T}) = d(v_i, \mathcal{T}) + d(v, \mathcal{T}_i)$.
15:     end if
16: end for
17: Release $d(v, \mathcal{T})$ for all $v \in \mathcal{V}$.
```

roughly $O(\log^3 V)/\varepsilon$ it suffices to show that any distance in $\mathcal{T}$ can be represented as a sum of $O(\log V)$ of the noisy distances released in the algorithm. We will use Lemma 5.3.1 to obtain a slightly better bound on the error terms.

Let set Q consist of the pairs of vertices corresponding to distance queries made by the algorithm above. We prove the following statement by induction. For any vertex $u \in \mathcal{V}$, there is a path from v_0 to u in the graph $(\mathcal{V}, Q)$ consisting of at most $2 \log V$ edges. The base case $V = 1$ is vacuous. For $V > 1$, note that since subtrees $T_0, \dots, T_t$ partition the vertex set, u must lie in one of them. If $u \in T_i$, then by the inductive assumption on T_i, there is a path from v_i to u in (V, Q) consisting of at most $2\log(V/2) = 2\log V - 2$ edges. If $i = 0$ this already suffices. Otherwise, if $i > 0$, note that $(v_0, v^*) \in Q$ and $(v^*, v_i) \in Q$. Consequently there is a path in $(\mathcal{V}, Q)$ from v_0 to u consisting of at most $2 \log V$ edges.

This means that there is a set of at most $2 \log V$ distances queried such that the distance from v_0 to u consists of the sum of these distances. Consequently by adding these distances we obtain an estimate for the distance from v_0 to u whose error is the sum of at most $2 \log V$

independent random variables each distributed according to $\mathsf{Lap}(\log V/\varepsilon)$. By Lemma 5.3.1, we have that with probability at least $1-\gamma$, we compute an estimate of $d_w(v_0, u)$ with error $O(\log^{1.5} V \cdot \log(1/\gamma))/\varepsilon$ for any $\gamma \in (0,1)$. Since differentially private mechanism are preserved under post-processing, this algorithm satisfies ε-differential privacy and computes distances from v^* to each other vertex in T, where with probability at least $1-\gamma$ each distance has error at most $O(\log^{1.5} V \cdot \log(1/\gamma))/\varepsilon$. ☐

We now extend this result to obtain all-pairs distances for trees.

Theorem 5.4.2 (All-pairs distances on trees). *For any tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ and $\varepsilon > 0$ there is an algorithm that is ε-differentially private on $\mathcal{T}$ and on input edge weights $w : \mathcal{E} \to \mathbb{R}^+$ releases all-pairs distances such that with probability $1-\gamma$, all released distances have approximation error*

$$O(\log^{2.5} V \cdot \log(1/\gamma))/\varepsilon$$

for any $\gamma \in (0,1)$. For each released distance, with probability $1-\gamma$ the approximation error is $O(\log^{1.5} V \cdot \log(1/\gamma))/\varepsilon$.

Proof. Arbitrarily choose some root vertex v_0. Use the ε-differentially private algorithm of Theorem 5.4.1 to obtain approximate distances from v_0 to each other vertex of $\mathcal{T}$. We now show that this suffices to obtain all-pairs distances.

Consider any pair of vertices x, y, and let z be their lowest common ancestor in the tree rooted at v_0. Then

$$\begin{aligned} d_w(x,y) &= d_w(z,x) + d_w(z,y) \\ &= d_w(v_0,x) + d_w(v_0,y) - 2d_w(v_0,z). \end{aligned}$$

Since with probability $1-3\gamma$ we can compute each of these three distances with error $O(\log^{1.5} V \cdot \log(1/\gamma)/\varepsilon)$, it follows that we can compute the distance between x and y with error at most four times this, which is still $O(\log^{1.5} V \cdot \log(1/\gamma))/\varepsilon$. By a union bound, for any $\gamma \in (0,1)$, with probability at least $1-\gamma$ each error among the $V(V-1)/2$ all-pairs distances released is at most $O(\log^{1.5} V \cdot \log(V/\gamma))/\varepsilon = O(\log^{2.5} V \cdot \log(1/\gamma))/\varepsilon$. ☐

5.4.2 Distances in bounded-weight graphs

Theorem 5.4.3. *For all $\mathcal{G}, \delta, \gamma, M$, and $\varepsilon \in (0,1)$, if $1/V < M\varepsilon < V$ then there is an algorithm $\mathcal{A}$ that is (ε, δ)-differentially private on $\mathcal{G}$ such that for all $w : \mathcal{E} \to [0, M]$, with probability $1-\gamma$, $\mathcal{A}(w)$ outputs approximate all-pairs distances with additive error*

$$O\left(\sqrt{VM\varepsilon^{-1} \cdot \log(1/\delta)} \cdot \log(VM\varepsilon/\gamma)\right)$$

per distance. For any $\varepsilon > 0$, $\mathcal{G}, \gamma, M$, if $1/V < M\varepsilon < V^2$ then there is an algorithm $\mathcal{B}$ that is ε-differentially private on $\mathcal{G}$ such that for all $w : \mathcal{E} \to [0, M]$, with probability $1-\gamma$, $\mathcal{B}(w)$ outputs approximate all-pairs distances with additive error

$$O\left((VM)^{2/3}\varepsilon^{-1/3}\log(VM\varepsilon/\gamma)\right)$$

per distance.

Algorithm 9 Bounded-weight distances

```
1: Inputs: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $w : \mathcal{E} \to \mathbb{R}^+$, $k, M, \gamma, \varepsilon' > 0$, $k$-covering $\mathcal{Z}$.
2: for $y, z \in \mathcal{Z}$ do
3:     Sample $X_{y,z} \leftarrow \mathsf{Lap}(|\mathcal{Z}|/\varepsilon')$ and let $a_{y,z} := d_w(y, z) + X_{y,z}$.
4: end for
5: For $v \in V$, let $z(v) \in \mathcal{Z}$ denote a vertex in $\mathcal{Z}$ with $h(v, z(v)) \leq k$.
6: The approximate distance between vertices $u, v \in V$ is given by $a_{z(u),z(v)}$.
```

To achieve this result, we will find a small subset $\mathcal{Z}$ of $\mathcal{V}$ such that every vertex $v \in \mathcal{V}$ is near some vertex $z \in \mathcal{Z}$. We will need the following definition, introduced in [MM75].

Definition 5.4.1. *A subset $\mathcal{Z} \subset \mathcal{V}$ of vertices is a **k-covering** if for every $v \in \mathcal{V}$ there is some $z \in \mathcal{Z}$ such that $h(v, z) \leq k$, where h is the hop-distance.*

A k-covering is sometimes called a k-dominating set (e.g. in [CN82]). The following lemma shows that we can find a sufficiently small k-covering for any graph $\mathcal{G}$.

Lemma 5.4.4. *[MM75] If $V \geq k+1$, then $\mathcal{G}$ has a k-covering of size at most $\lfloor V/(k+1) \rfloor$.*

Proof. Consider any spanning tree $\mathcal{T}$ of $\mathcal{G}$ and any vertex $x \in \mathcal{V}$ that is an endpoint of one of the longest paths in $\mathcal{T}$. For $0 \leq i \leq k$, let $\mathcal{Z}_i$ be the subset of $\mathcal{V}$ consisting of vertices whose distance from x in $\mathcal{T}$ is congruent to i modulo $k+1$. It can be shown that each $\mathcal{Z}_i$ is a k-covering of $\mathcal{T}$ and therefore of $\mathcal{G}$. But the $k+1$ sets $\mathcal{Z}_i$ form a partition of $\mathcal{V}$. Therefore some $\mathcal{Z}_i$ has $|\mathcal{Z}_i| \leq \lfloor V/(k+1) \rfloor$, as desired. $\square$

Theorem 5.4.5. *For any $\mathcal{G} = (\mathcal{V}, \mathcal{E}), k, \delta > 0$, and $\gamma, \varepsilon \in (0,1)$, let $\mathcal{Z}$ be a k-covering of $\mathcal{G}$, and let $Z = |\mathcal{Z}|$. Then there is an algorithm $\mathcal{A}$ which is (ε, δ)-differentially private on $\mathcal{G}$ such that for any edge weight function $w : \mathcal{E} \to [0, M]$, with probability $1 - \gamma$, $\mathcal{A}(w)$ releases all-pairs distances with approximation error*

$$O\left(kM + Z\varepsilon^{-1} \log(Z/\gamma)\sqrt{\log(1/\delta)}\right)$$

per distance.

Proof. There are Z^2 pairwise distances between vertices in $\mathcal{Z}$. We can compute and release noisy versions of each of these distances, adding $\mathsf{Lap}(Z/\varepsilon')$ noise to each. For any $\gamma \in (0,1)$, with probability $1-\gamma$ we have that each of these Z^2 noise variables has magnitude at most $(Z/\varepsilon')\log(Z^2/\gamma)$. By Lemma 5.3.4, for any $\delta > 0$ releasing this is (ε, δ)-differentially private for $\varepsilon' = O(\varepsilon/\sqrt{\ln 1/\delta}\,)$.

But this information allows the recipient to compute approximate distances between any pair of vertices $x, y \in \mathcal{V}$, as follows. Since $\mathcal{Z}$ is a k-covering of $\mathcal{G}$, we can find $z_x, z_y \in \mathcal{Z}$ which are at most k vertices from x and y. Since the maximum weight is M, the weight of the shortest path between x and z_x is at most kM, and similarly for y and z_y. Consequently

$$|d_w(x,y) - d_w(z_x, z_y)| \leq 2kM.$$

But we have released an estimate of $d_w(z_x, z_y)$ with noise distributed according to $\mathsf{Lap}(Z/\varepsilon)$. Consequently with probability $1-\gamma$ each of these estimates differs from $d_w(z_x, z_y)$ by at most $(Z/\varepsilon)\log(Z^2/\gamma)$. $\square$

We obtain a slightly weaker result under pure differential privacy.

Theorem 5.4.6. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For any $k > 0$ and $\gamma \in (0,1)$, if $\mathcal{Z}$ is a k-covering of $\mathcal{G}$ of size Z, then there is an algorithm $\mathcal{A}$ that is ε-differentially private on $\mathcal{G}$ such that for any $w : \mathcal{E} \to [0, M]$, with probability $1-\gamma$, $\mathcal{A}(w)$ releases all-pairs distances with approximation error*

$$O(kM + Z^2\varepsilon^{-1}\log(Z/\gamma))$$

per distance.

Proof. There are at most Z^2 pairwise distances between vertices in $\mathcal{Z}$, so we can release approximations of each distance, adding $\mathsf{Lap}(Z^2/\varepsilon)$ noise to each distance. With probability $1-\gamma$ each of these Z^2 noise variables has magnitude at most $(Z^2/\varepsilon)\log(Z^2/\gamma)$. By Lemma 5.3.3, releasing these distances is ε-differentially private. As above, since $\mathcal{Z}$ is a k-covering of $\mathcal{G}$, we can find $z_x, z_y \in \mathcal{Z}$ which are at most k vertices from x and y. Consequently $d_w(x, z_x) \leq kM$ and $d_w(y, z_y) \leq kM$, so

$$|d_w(x,y) - d_w(z_x, z_y)| \leq 2kM.$$

But the released estimate for $d_w(z_x, z_y)$ has noise distributed according to $\mathsf{Lap}(kM)$, so with probability $1-\gamma$ each of these estimates differs from $d_w(x,y)$ by at most $O(Z^2\varepsilon^{-1}\log(Z^2/\gamma))$, as desired. □

We now conclude the proof of Theorem 5.4.3.

Proof of Theorem 5.4.3. Combine Lemma 5.4.4 with Theorem 5.4.5 for $k = \lfloor\sqrt{V/(M\varepsilon)}\rfloor$, and combine Lemma 5.4.4 with Theorem 5.4.6 for $k = \lfloor V^{2/3}/(M\varepsilon)^{1/3}\rfloor$. □

Note that if we are only interested in the $V-1$ distances from a single source, then directly releasing noisy distances and applying Lemma 5.3.4 yields (ε,δ)-differential privacy with error distributed according to $\mathsf{Lap}(b)$ for $b = O(\sqrt{V\log 1/\delta})/\varepsilon$, which has the same dependence on V as the bound provided by the theorem for releasing all pairs distances.

For some graphs we may be able to find a smaller k-covering than that guaranteed by Lemma 5.4.4. Then we can use Theorems 5.4.5 and 5.4.6 to obtain all-pairs distances with lower error. For instance, we have the following.

Theorem 5.4.7. *Let $\mathcal{G}$ be the $\sqrt{V} \times \sqrt{V}$ grid. Then for any $\varepsilon, \gamma \in (0,1)$ and $\delta > 0$, we can release with probability $1 - \gamma$ all-pairs distances with additive approximation error*

$$V^{1/3} \cdot O\left(M + \varepsilon^{-1} \log(V/\gamma)\sqrt{\log 1/\delta}\right)$$

while satisfying (ε, δ)-differential privacy.

Proof. Let $\mathcal{V} = [\sqrt{V}] \times [\sqrt{V}]$, and let $\mathcal{Z} \subset \mathcal{V}$ consist of vertices $(i,j) \in \mathcal{V}$ with i, j both one less than a multiple of $V^{1/3}$. Then $\mathcal{Z}$ is a $2V^{1/3}$-covering of $\mathcal{G}$, and also $|\mathcal{Z}| \leq V^{1/3}$. Consequently Theorem 5.4.5 implies the desired conclusion. □

5.5 Finding shortest paths

5.5.1 Lower bound

In this section we present a lower bound on the additive error with which we can privately release a short path between a pair of vertices. The argument is based on a reduction from the problem of reconstructing a large fraction of the entries of a database. We show that an adversary can use an algorithm which outputs a short path in a graph to produce a vector with small Hamming distance to an input vector, which is impossible under differential privacy. To that end, we exhibit a "hard" graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a family of weight functions, and provide a correspondence between inputs $x \in \{0,1\}^n$ and weight functions $w : \mathcal{E} \to \{0,1\}$.

Theorem 5.5.1. *There exists a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and vertices $s, t \in \mathcal{V}$ such that for any algorithm $\mathcal{A}$ that is (ε, δ)-differentially private on $\mathcal{G}$, there exist edge weights $w : \mathcal{E} \to \{0,1\}$ for which the expected approximation error of the path $\mathcal{A}(w)$ from s to t is at least $\alpha = (V-1) \cdot \left(\frac{1-(1+e^{\varepsilon})\delta}{1+e^{2\varepsilon}}\right)$. In particular, for sufficiently small ε and δ, $\alpha \geq 0.49(V-1)$.*

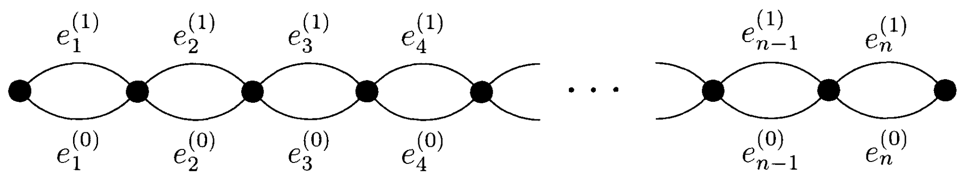


Figure 5-2: The graph used for the lower bound of Lemma 5.5.2.

Consequently, any differentially private algorithm for approximate shortest paths must on some inputs have additive error proportional to the number of vertices. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the $(n+1)$-vertex graph with vertex set $\mathcal{V} = \{0, \dots, n\}$ and two parallel edges $e_i^{(0)}$ and $e_i^{(1)}$ between each pair of consecutive vertices $i-1, i$, as shown in Figure 5-2. (For simplicity we have defined this is a multigraph, but it can be transformed into a simple graph with the addition of n extra vertices, changing the bound obtained by a factor of 2.)

Lemma 5.5.2. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph defined in the previous paragraph. For any α, let $\mathcal{A}$ be an algorithm that is (ε, δ)-differentially private on $\mathcal{G}$ that on input edge weights $w : \mathcal{E} \to \{0, 1\}$ produces a path from vertex $s = 0$ to vertex $t = n$ with expected approximation error at most α. Then there exists a $(2\varepsilon, (1 + e^{\varepsilon})\delta)$-differentially private algorithm $\mathcal{B}$ which on input $x \in \{0, 1\}^n$ produces $y \in \{0, 1\}^n$ such that the expected Hamming distance $d_H(x, y)$ is at most α.*

Proof. Given an input $x \in \{0, 1\}^n$, the corresponding edge weight function w_x is given by $w_x(e_i^{(x_i)}) = 0$ and $w_x(e_i^{(1-x_i)}) = 1$. That is, for each pair of consecutive vertices $i-1, i$, one of the edges between them will have weight 0 and the other will have weight 1 as determined by the ith bit of the input x.

The algorithm $\mathcal{B}$ is as follows. On input $x \in \{0, 1\}^n$, apply $\mathcal{A}$ to $(\mathcal{G}, w_x)$, and let $\mathcal{P}$ be the path produced. Define $y \in \{0, 1\}^n$ as follows. Let $y_i = 0$ if $e_i^{(0)} \in \mathcal{P}$ and $y_i = 1$ otherwise. Output y.

We first show that this procedure is differentially private. Given neighboring inputs $x, x' \in \{0, 1\}^n$ which differ only on a single coordinate $x_i \neq x'_i$, we have that the associated weight functions w_x and $w_{x'}$ have ℓ_1 distance 2, since they disagree only edges $e_i^{(0)}$ and $e_i^{(1)}$.

Consequently, since $\mathcal{A}$ is (ε, δ)-differentially private, we have that for any set of values S in the range of $\mathcal{A}$,

$$\begin{aligned}\Pr[\mathcal{A}(w_x) \in S] &\leq e^{\varepsilon}(e^{\varepsilon} \Pr[\mathcal{A}(w_{x'}) \in S] + \delta) + \delta \\ &= e^{2\varepsilon} \Pr[\mathcal{A}(w_{x'}) \in S] + (1 + e^{\varepsilon})\delta \ .\end{aligned}$$

But algorithm $\mathcal{B}$ only accesses the database x through $\mathcal{A}$. Consequently by the robustness of differential privacy to post-processing, we have that $\mathcal{B}$ is $(2\varepsilon, (1 + e^{\varepsilon})\delta)$-differentially private.

We now show that the expected number of coordinates in which y disagrees with x is at most α. The shortest path from s to t in $\mathcal{G}$ has length 0, so the expected length of the path $\mathcal{P}$ produced by $\mathcal{A}$ is at most α. Consequently in expectation $\mathcal{P}$ consists of at most α edges $e_i^{(1-x_i)}$. But $y_i \neq x_i$ only if $e_i^{(1-x_i)} \in \mathcal{P}$, so it follows that the expected Hamming distance $d_H(x, y) \leq \alpha$. □

We will now prove two simple and standard lemmas concerning the limits of identifying rows of the input of a differentially private algorithm. In the first lemma, we show that a differentially private algorithm cannot release a particular row of its input with probability greater than $\frac{e^{\varepsilon}+\delta}{1+e^{\varepsilon}}$. In essence, for $\delta = 0$ this can be interpreted as a statement about the optimality of the technique of randomized response [War65].

Lemma 5.5.3. *If algorithm $\mathcal{B} : \{0,1\}^n \to \{0,1\}$ is (ε, δ)-differentially private, then if we uniformly sample a random input $X \leftarrow U_n$, we have that for all i,*

$$\Pr[\mathcal{B}(X) \neq X_i] \geq \frac{1 - \delta}{1 + e^{\varepsilon}}$$

Proof. We have that $\Pr[\mathcal{B}(X) = X_i]$ is given by

$$\begin{aligned}&\frac{1}{2} \Pr[\mathcal{B}(X_{-i}, 0) = 0] + \frac{1}{2} \Pr[\mathcal{B}(X_{-i}, 1) = 1] \\ &\leq \frac{e^{\varepsilon}}{2} \cdot (\Pr[\mathcal{B}(X_{-i}, 1) = 0] + \Pr[\mathcal{B}(X_{-i}, 0) = 1]) + \delta \\ &= e^{\varepsilon} \Pr[\mathcal{B}(X) \neq X_i] + \delta\end{aligned}$$

so since $\Pr[\mathcal{B}(X) = X_i] = 1 - \Pr[\mathcal{B}(X) \neq X_i]$,

$$\Pr[\mathcal{B}(X) \neq X_i] \geq \frac{1-\delta}{1+e^{\varepsilon}} .$$

□

This immediately implies the following result.

Lemma 5.5.4. *If algorithm $\mathcal{B} : \{0,1\}^n \to \{0,1\}^n$ is (ε, δ)-differentially private, then for some $x \in \{0,1\}^n$ we have that the expected Hamming distance $d_H(\mathcal{B}(x), x)$ is at least $\frac{n(1-\delta)}{1+e^{\varepsilon}}$.*

Proof. By Lemma 5.5.3, projecting onto any coordinate i, the probability that $\Pr[\mathcal{B}(X)_i \neq X_i]$ for uniformly random $X \leftarrow U_n$ is at least $\frac{1-\delta}{1+e^{\varepsilon}}$. Consequently the expected number of coordinates on which $\mathcal{B}(X)$ differs from X is $\mathbf{E}(d_H(\mathcal{B}(X), X)) \geq \frac{n(1-\delta)}{1+e^{\varepsilon}}$. Since this holds for X uniformly random, in particular we have that there exists some $x \in \{0,1\}^n$ such that

$$\mathbf{E}(d_H(\mathcal{B}(x), x)) \geq \frac{n(1-\delta)}{1+e^{\varepsilon}}.$$

□

We now conclude the proof of Theorem 5.5.1.

Proof of Theorem 5.5.1. Assume that there is some algorithm which is (ε, δ)-differentially private on $\mathcal{G}$ and always produces a path of error at most α between s and t. Then by Lemma 5.5.2 we have that there is a $(2\varepsilon, (1+e^{\varepsilon})\delta)$-differentially private algorithm $\mathcal{B}$ which for all $x \in \{0,1\}^n$ produces $y \in \{0,1\}^n$ such that the expected Hamming distance $d_H(x, y)$ is less than α. But by Lemma 5.5.4, for some x the expected Hamming distance $\mathbf{E}(d_H(x,y)) \geq \frac{n(1-(1+e^{\varepsilon})\delta)}{1+e^{2\varepsilon}} = \alpha$, yielding a contradiction. □

5.5.2 Upper bound

In this section we show that an extremely simple algorithm matches the lower bound of the previous section up to a logarithmic factor, for fixed ε, δ. Consider a direct application of the

Laplace mechanism (Lemma 5.3.2), adding $\mathsf{Lap}(1/\varepsilon)$ noise to each edge weight and releasing the resulting values. With high probably all of these $E < V^2$ noise variables will be small, providing a bound on the difference in the weight of any path between the released graph and the original graph. Consequently we can show that if we take the shortest path in the released graph, with 99% probability the length of the same path in the original graph is $O(V \log V)/\varepsilon$ longer than optimal.

This straightforward application of the Laplace mechanism almost matches the lower bound of the previous section. Surprisingly, with the same error bound it releases not just a short path between a single pair of vertices but short paths between all pairs of vertices.

One drawback of this argument is that the error depends on the size of the entire graph. In practice we may expect that the shortest path between most pairs of vertices consist of relatively few edges. We would like the error to depend on the number of hops on the shortest path rather than scaling with the number of vertices. We achieve this with a post-processing step that increases the weight of all edges, introducing a preference for few-hop paths. We show that if there is a short path with only k hops, then our algorithm reports a path whose length is at most $O(k \log V/\varepsilon)$ longer.

Theorem 5.5.5. *For all graphs $\mathcal{G}$ and $\gamma \in (0,1)$, Algorithm 10 is ε-differentially private on $\mathcal{G}$ and computes paths between all pairs of vertices such that with probability $1-\gamma$, for all pairs of vertices $s,t \in \mathcal{V}$, if there exists a k-hop path of weight W in $(\mathcal{G}, w)$, the path released has weight at most $W + (2k/\varepsilon)\log(E/\gamma)$.*

In particular, if the shortest path in $\mathcal{G}$ has k hops, then Algorithm 10 releases a path only $(2k/\varepsilon)\log(E/\gamma)$ longer than optimal. This error term is proportional to the number of hops on the shortest path. Noting that the shortest path between any pair of vertices consists of fewer than V hops, we obtain the following corollary.

Corollary 5.5.6. *For any $\gamma \in (0,1)$, with probability $1-\gamma$ Algorithm 10 computes paths between every pair of vertices with approximation error at most $(2V/\varepsilon)\log(E/\gamma)$.*

Algorithm 10 Private shortest paths

```
1: Inputs: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $w : \mathcal{E} \to \mathbb{R}^+$, $\gamma > 0$.
2: for each edge $e \in \mathcal{E}$ do
3:     Sample $X_e \leftarrow \mathsf{Lap}(1/\varepsilon)$
4:     Let $w'(e) = w(e) + X_e + (1/\varepsilon)\log(E/\gamma)$, where $E = |\mathcal{E}|$
5: end for
6: Output $w'$.
7: The approximate shortest path between a pair of vertices $x, y \in \mathcal{V}$ is taken to be the
   shortest path $\mathrm{SP}_{w'}(x, y)$ in the weighted graph $(\mathcal{G}, w')$.
```

Proof of Theorem 5.5.5. Each random variable X_e is distributed according to $\mathsf{Lap}(1/\varepsilon)$, so with probability $1 - \gamma$ we have that $|X_e| \leq (1/\varepsilon)\log(1/\gamma)$ for any $\gamma \in (0, 1)$. By a union bound, with probability $1 - \gamma$ all $E < V^2$ of these random variables have magnitude at most $(1/\varepsilon)\log(E/\gamma)$. Conditioning on this event, for each edge $e \in \mathcal{E}$, the modified weight computed by the algorithm satisfies

$$w(e) \leq w'(e) \leq w(e) + (2/\varepsilon)\log(E/\gamma).$$

Therefore, for any k-hop path $\mathcal{P}$ we have that

$$w(\mathcal{P}) \leq w'(\mathcal{P}) \leq w(\mathcal{P}) + (2k/\varepsilon)\log(E/\gamma).$$

For any $s, t \in \mathcal{V}$, if $\mathcal{Q}$ is the path from s to t produced by the algorithm and $\mathcal{Q}'$ is any path from s to t, then we have that $\mathcal{Q}$ is a shortest path under w', so

$$w(\mathcal{Q}) \leq w'(\mathcal{Q}) \leq w'(\mathcal{Q}') \leq w(\mathcal{Q}') + (2\ell(\mathcal{Q}')/\varepsilon)\log(E/\gamma).$$

□

5.6 Distances in the path graph

In this section we give an explicit description of the private all-pairs distance algorithm for the path graph $\mathcal{P} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = [V]$ and edge set $\mathcal{E} = \{(i, i+1) : i \in [V-1]\}$. This result is a restatement of a result of [DNPR10], and is generalized to trees in Section 5.4.1. This alternate argument for the special case of the path graph is included for illustration.

The idea behind the construction is as follows. We designate a small set of hubs and store more accurate distances between consecutive hubs. As long as we can accurately estimate the distance from any vertex to the nearest hub and the distance between any pair of hubs, we can use these distances to obtain an estimate of the distance between any pair of vertices. Given any pair of vertices $x, y \in \mathcal{V}$, in order to estimate the distance $dist(x, y)$, we first find the hubs h_x and h_y nearest to x and y. We estimate the distance $dist(x, y)$ by adding our estimates for the distances $dist(x, h_x)$, $dist(h_x, h_y)$, and $dist(h_y, y)$.

Instead of simply using a single set of hubs, we will use a hierarchical tree of hubs of different levels. There will be many hubs of low level and only a small number of hubs of high level. Each hub will store an estimate of the distance to the next hub at the same level and every lower level. Hubs higher in the hierarchy will allow us to skip directly to distant vertices instead of accruing error by adding together linearly many noisy estimates. In order to estimate the distance between a particular pair of vertices $x, y \in \mathcal{V}$, we will only consider a small number of hubs on each level of the hierarchy. Since the total number of levels is not too large, this will result in a much more accurate differentially private estimate of the distance between x and y.

Theorem 5.6.1. *Let $\mathcal{P} = (\mathcal{V}, \mathcal{E})$ be the path graph on V vertices. For any $\varepsilon > 0$, there is algorithm $\mathcal{A}$ that is ε-differentially private on $\mathcal{P}$ that on input $w : \mathcal{E} \to \mathbb{R}^+$ releases approximate all-pairs distances such that for each released distance, with probability $1 - \gamma$ the approximation error is $O(\log^{1.5} V \log(1/\gamma))/\varepsilon$ for any $\gamma \in (0, 1)$.*

Proof. For fixed k, define nested subsets $\mathcal{V} = S_0 \supset S_1 \supset \ldots \supset S_{k-1}$ of the vertex set $\mathcal{V} = [V]$ as follows. Let

$$S_i = \{x \in [V] : V^{i/k} \mid x\}.$$

That is, S_0 consists of all the vertices, and in general S_i consists of one out of every $V^{i/k}$ vertices on the path. Then $|S_i| = V^{(k-i)/k}$. Let $s_{i,1}, s_{i,2} \dots, s_{i,|S_i|}$ denote the elements of S_i in increasing order, for each i. Using the Laplace mechanism (Lemma 5.3.2), release noisy distances between each pair of consecutive vertices $s_{i,j}, s_{i,j+1}$ of each set, adding noise proportional to $\mathsf{Lap}(k/\varepsilon)$. Note that since the vertices in each S_i are in increasing order, each edge of P is only considered for a single released difference from each set. Consequently the total sensitivity to release all of these distances is k, so releasing these noisy distances is ε-differentially private. Using post-processing and these special distances, we will compute approximate all-pairs distances with small error.

For any pair of vertices x, y, consider the path $\mathcal{P}[x, y]$ in $\mathcal{P}$ between x and y, and let i be the largest index such that S_i contains multiple vertices of $\mathcal{P}[x, y]$. We must have that $S_i \cap \mathcal{P}[x, y] < 2V^{1/k}$, since otherwise S_{i+1} would contain at least two vertices on $\mathcal{P}[x, y]$. Let x_i, y_i denote the first and last vertices in $S_i \cap \mathcal{P}[x, y]$. For $j < i$, let x_j denote the first vertex in $S_j \cap \mathcal{P}[x, x_i]$ and let y_j denote the last vertex in $S_j \cap \mathcal{P}[y_i, y]$. There are at most $1 + V^{1/k}$ vertices of S_j in $\mathcal{P}[x_j, x_{j+1}]$, since otherwise this interval would contain another vertex of S_{j+1}. Similarly, there are at most $1 + V^{1/k}$ vertices of S_j in $\mathcal{P}[y_{j+1}, y_j]$. Therefore we can express the distance from x_j to x_{j+1} as the sum of at most $V^{1/k}$ distances which were estimated in S_j, and similarly for the distance from y_{j+1} to y_j.

Putting this all together, we can estimate the distance from $x = x_0$ to $y = y_0$ as the sum of at most $2(i+1)V^{1/k} \leq 2kV^{1/k}$ approximate distances which were released. But each of these distances is released with noise distributed according to $\mathsf{Lap}(k/\varepsilon)$. Consequently the total error on the estimated distance from x to y is the sum of at most $2kV^{1/k}$ random variables distribued according to $\mathsf{Lap}(k/\varepsilon)$. Taking $k = \log V$, the error is the sum of at most $4 \log V$ variables distributed according to $\mathsf{Lap}(\log V/\varepsilon)$. By Lemma 5.3.1, with probability at least $1 - \gamma$ the sum of these $4 \log V$ variables is bounded by $O(\log^{1.5} V \log(1/\gamma))/\varepsilon$ for any $\gamma \in (0, 1)$. Consequently this is an ε-differentially private algorithm which releases all-pairs distances in the path graph P such that for any $\gamma \in (0, 1)$, with probability at least $1 - \gamma$ the error in each released distance is at most $O(\log^{1.5} V \log(1/\gamma))/\varepsilon$. □

5.7 Other graph problems

In this section we consider some additional queries on graphs in the private edge weights model.

5.7.1 Almost-minimum spanning tree

We first consider the problem of releasing a low-cost spanning tree. The work of [NRS07] showed how to privately approximate the cost of the minimum spanning tree in a somewhat related privacy setting. We seek to release an actual tree of close to minimal cost under our model. Using techniques similar to the lower bound for shortest paths from Section 5.5.1, we obtain a lower bound of $\Omega(V)$ for the error of releasing a low-cost spanning tree, and show that the Laplace mechanism yields a spanning tree of cost $O(V \log V)$ more than optimal. Note that in this section edge weights are permitted to be negative.

Theorem 5.7.1. *There exists a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that for any spanning tree algorithm $\mathcal{A}$ that is (ε, δ)-differentially private on $\mathcal{G}$, there exist edge weights $w : \mathcal{E} \to \{0, 1\}$ such that the expected weight of the spanning tree $\mathcal{A}(w)$ is at least*

$$\alpha = (V - 1) \cdot \left(\frac{1 - (1 + e^{\varepsilon})\delta}{1 + e^{2\varepsilon}} \right)$$

longer than the weight of the minimum spanning tree. In particular, for sufficiently small ε and δ, $\alpha \geq 0.49(V - 1)$.

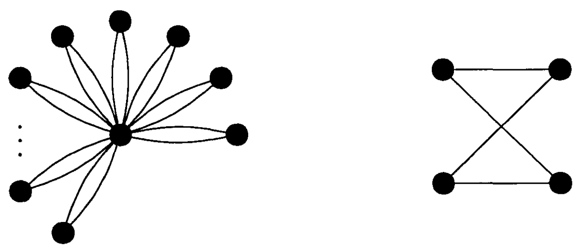


Figure 5-3: (Left) The graph used in the reduction of Lemma 5.7.2. (Right) A single gadget in the graph used in the reduction of Lemma 5.7.5.

We first prove a lemma reducing the problem of reidentifying rows in a database to privately finding an approximate minimum spanning tree. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the $(n+1)$-vertex graph with vertex set $\mathcal{V} = \{0, \dots, n\}$ and a pair of edges $e_i^{(0)}$ and $e_i^{(1)}$ between vertex 0 and each vertex $i > 0$, as shown in Figure 5-3. (As in Lemma 5.5.2, this is a multigraph, but we can transform it into a simple graph by adding n extra vertices, changing the bound obtained by a factor of 2.)

Lemma 5.7.2. *Let $\mathcal{G}$ be the graph defined in the previous paragraph, and let $\varepsilon, \delta \geq 0$. For any α, let $\mathcal{A}$ be an algorithm that is (ε, δ)-differentially private on $\mathcal{G}$ that on input $w : \mathcal{E} \to \{0,1\}$ produces a spanning tree whose weight in expectation is at most α greater than optimal. Then there exists a $(2\varepsilon, (1+e^{\varepsilon})\delta)$-differentially private algorithm $\mathcal{B}$ which on input $x \in \{0,1\}^n$ produces $y \in \{0,1\}^n$ such that the expected Hamming distance $d_H(x,y)$ is at most α.*

Proof. The outline of the proof is the same as that of Lemma 5.5.2. For input $x \in \{0,1\}^n$, the corresponding edge weight function w_x is given by $w_x(e_i^{(x_i)}) = 0$ and $w_x(e_i^{(1-x_i)}) = 1$, where x_i is the ith bit of x.

We define algorithm $\mathcal{B}$ as follows. On input $x \in \{0,1\}^n$, apply $\mathcal{A}$ to $(\mathcal{G}, w_x)$, and let $\mathcal{T}$ be the tree produced. Define $y \in \{0,1\}^n$ by setting $y_i = 0$ if $e_i^{(0)} \in \mathcal{T}$ and $y_i = 1$ otherwise. Output y.

It is straightforward to verify that $\mathcal{B}$ is $(2\varepsilon, (1+e^{\varepsilon})\delta)$-differentially private. We now bound the expected Hamming distance of x and y. The minimum spanning tree in $\mathcal{G}$ has weight 0, so the expected weight of $\mathcal{T}$ is at most α and $\mathcal{T}$ must consist of at most α edges $e_i^{(1-x_i)}$. But $y_i \neq x_i$ only if $e_i^{(1-x_i)} \in \mathcal{T}$, so in expectation $d_H(x,y) \leq w(\mathcal{T}) \leq \alpha$. $\square$

We now complete the proof of Theorem 5.7.1.

Proof of Theorem 5.7.1. Assume that there is some (ε, δ)-differentially private algorithm which on all inputs produces a spanning tree with expected weight less than α more than optimal. By Lemma 5.7.2, there is a $(2\varepsilon, (1+e^{\varepsilon})\delta)$-differentially private algorithm which for all $x \in \{0,1\}^n$ produces $y \in \{0,1\}^n$ with expected Hamming distance less than α. But then

Lemma 5.5.4, for some x the expected Hamming distance $\mathbf{E}(d_H(x,y)) \geq \frac{n(1-(1+e^{\varepsilon})\delta)}{1+e^{2\varepsilon}} = \alpha$, yielding a contradiction. □

We now show that the Laplace mechanism (Lemma 5.3.2) almost matches this lower bound.

Theorem 5.7.3. *For any $\varepsilon, \gamma > 0$ and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, there is an algorithm $\mathcal{A}$ that is ε-differentially private on $\mathcal{G}$ that on input $w : \mathcal{E} \to \mathbb{R}$ releases with probability $1-\gamma$ a spanning tree of weight at most $((V-1)/\varepsilon)\log(E/\gamma)$ larger than optimal.*

Proof. Consider the algorithm that adds noise X_e distributed according to $\mathsf{Lap}(1/\varepsilon)$ for each edge $e \in \mathcal{E}$ and releases the minimum spanning tree on the resulting graph $(\mathcal{G}, w')$. This is ε-differentially private, since it is post-processing of the Laplace mechanism. We now show that the resulting error is small. By a union bound, with probability $1-\gamma$ we have that $|X_e| \leq (1/\varepsilon)\log(E/\gamma)$ for every $e \in \mathcal{E}$. Consequently, conditioning on this event, if $\mathcal{T}$ is the spanning tree released by the algorithm and $\mathcal{T}^*$ is the minimum spanning tree, then we have that

$$\begin{aligned} w(\mathcal{T}) &\leq w'(\mathcal{T}) + \frac{V-1}{\varepsilon} \cdot \log(E/\gamma) \\ &\leq w'(\mathcal{T}^*) + \frac{V-1}{\varepsilon} \cdot \log(E/\gamma) \\ &\leq w(\mathcal{T}^*) + \frac{2(V-1)}{\varepsilon} \cdot \log(E/\gamma). \end{aligned}$$

□

5.7.2 Low-weight matching

We now consider the problem of releasing a minimum weight matching in a graph in our model. As for the minimum spanning tree problem, a minor modification of the lower bound for shortest paths from Section 5.5.1 yields a similar result. For comparison, [HHR$^+$14] use similar reconstruction techniques to obtain a lower bound for a matching problem on bipartite graphs in a somewhat different model in which all edge weights are in $[0,1]$ and neighboring graphs can differ on the weights of the edges incident to a single left vertex. We show a lower bound of $\Omega(V)$ for the error of releasing a low-weight matching tree, and show that the matching released by the Laplace mechanism has weight $O(V \log V)$ greater than optimal.

The theorems in this section are stated for the problem of finding a minimum weight perfect matching. We can also obtain identical results for the problem of finding a minimum weight matching which is not required to be perfect, and for the corresponding maximum weight matching problems. Our results apply to both bipartite matching and general matching. Note that in this section edge weights are permitted to be negative.

Theorem 5.7.4. *There exists a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that for any perfect matching algorithm $\mathcal{A}$ which is (ε, δ)-differentially private on $\mathcal{G}$, there exist edge weights $w : \mathcal{E} \to \{0,1\}$ such that the expected weight of the matching $\mathcal{A}(w)$ is at least*

$$\alpha = \left(\frac{V}{4}\right) \cdot \left(\frac{1-(1+e^{\varepsilon})\delta}{1+e^{2\varepsilon}}\right)$$

larger than the weight of the min-cost perfect matching. In particular, for sufficiently small ε, δ, $\alpha \geq 0.12 \cdot V$.

The following lemma reduces the problem of reidentification in a database to finding a low-cost matching. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the $4n$-vertex graph with vertex set $\mathcal{V} = \{(b_1, b_2, c) : b_1, b_2 \in \{0,1\}, c \in [n]\}$ and edges from $(0, b, c)$ to $(1, b', c)$ for every $b, b' \in \{0,1\}$, $c \in [n]$. That is, $\mathcal{G}$ consists of n disconnected hourglass-shaped gadgets as shown in Figure 5-3.

Lemma 5.7.5. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph defined in the previous paragraph. For any α, let $\mathcal{A}$ be an algorithm that is (ε, δ)-differentially private on $\mathcal{G}$ that on input $w : \mathcal{E} \to \{0,1\}$ produces a perfect matching of expected weight at most α greater than optimal. Then there exists a $(2\varepsilon, (1+e^{\varepsilon})\delta)$-differentially private algorithm $\mathcal{B}$ which on input $x \in \{0,1\}^n$ produces $y \in \{0,1\}^n$ with expected Hamming distance to x at most α.*

Proof. For any input $x \in \{0,1\}^n$, the corresponding weight function w_x assigns weight 1 to the edge connecting vertex $(0, 1, i)$ to $(1, 1 - x_i, i)$ for each $i \in [n]$, and assigns weight 0 to the other $3n$ edges. The algorithm $\mathcal{B}$ is as follows. On input $x \in \{0,1\}^n$, apply $\mathcal{A}$ to $(\mathcal{G}, w_x)$, and let $\mathcal{M}$ be the matching produced. Define $y \in \{0,1\}^n$ as follows. Let $y_i = 0$ if the edge from $(0,1,i)$ to $(1,0,i)$ is in the matching, and $y_i = 1$ otherwise. Output y.

Algorithm $\mathcal{B}$ is clearly $(2\varepsilon, (1+e^{\varepsilon})\delta)$-differentially private. We will have that $y_i \neq x_i$ only if the edge from $(0,1,i)$ to $(1,1-x_i,i)$ is in the matching, so the expected Hamming distance is at most the expected size of the matching produced by $\mathcal{A}$, which is at most α. □

We now conclude the proof of Theorem 5.7.4.

Proof of Theorem 5.7.4. The result follows by combining Lemma 5.7.5 with Lemma 5.5.4. □

Using the Laplace mechanism (Lemma 5.3.2), we obtain a nearly-matching upper bound.

Theorem 5.7.6. *For any $\varepsilon, \gamma > 0$ and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ containing a perfect matching, there is an algorithm $\mathcal{A}$ that is ε-differentially private on $\mathcal{G}$ that on input $w : \mathcal{E} \to \mathbb{R}$ releases with probability $1-\gamma$ a perfect matching of weight at most $(V/\varepsilon)\log(E/\gamma)$ larger than optimal.*

Proof. Consider the algorithm that adds noise X_e distributed according to $\mathsf{Lap}(1/\varepsilon)$ for each edge $e \in \mathcal{E}$ and releases the minimum-weight perfect matching on the resulting graph $(\mathcal{G}, w')$. This is ε-differentially private, since it is post-processing of the Laplace mechanism. We now show that the resulting error is small. With probability $1-\gamma$ we have that $|X_e| \leq (1/\varepsilon)\log(E/\gamma)$ for every $e \in \mathcal{E}$. Consequently, conditioning on this event, if $\mathcal{M}$ is the matching released by the algorithm and $\mathcal{M}^*$ is the minimum-weight matching, then we have that

$$\begin{aligned} w(\mathcal{M}) &\leq w'(\mathcal{M}) + \frac{V}{2\varepsilon} \cdot \log(E/\gamma) \\ &\leq w'(\mathcal{M}^*) + \frac{V}{2\varepsilon} \cdot \log(E/\gamma) \\ &\leq w(\mathcal{M}^*) + \frac{V}{\varepsilon} \cdot \log(E/\gamma). \end{aligned}$$

□

Chapter 6

Efficiently estimating Erdős-Rényi graphs with node differential privacy

This chapter is based on joint work with Jon Ullman [SU19].

6.1 Introduction

Network data modeling individuals and relationships between individuals are increasingly central in data science. However, while there is a highly successful literature on differentially private statistical estimation for traditional iid data, the literature on estimating network models is far less well developed.

Early work on private network data focused on *edge-differential-privacy*, in which the algorithm is required to "hide" the presence or absence of a single edge in the graph (see, e.g. [NRS07, HLMJ09, KRSY14, GRU12, BBDS12, XCT14, KS16], and many others). A more desirable notion of privacy is *node-differential privacy (node-DP)*, which requires the algorithm to hide the presence or absence of an arbitrary set of edges incident on a single node. Although node-DP is difficult to achieve without compromising accuracy, the beautiful works of Blocki *et al.* [BBDS13] and Kasiviswanathan *et al.* [KNRS13] showed how to design accurate node-DP estimators for many interesting graph statistics via *Lips-*

chitz extensions. However, many of the known constructions of Lipschitz extensions require exponential running time, and constructions of computationally efficient Lipschitz extensions [RS16, CD18, CKM$^+$19] lag behind. As a result, even for estimating very simple graph models, there are large gaps in accuracy between the best known computationally efficient algorithms and the information theoretically optimal algorithms.

In this work we focus on what is arguably the simplest model of network data, the *Erdős-Rényi graph.* In this model, denoted $G(n,p)$, we are given a number of nodes n and a parameter $p \in [0,1]$, and we sample an n-node graph G by independently including each edge (i,j) for $1 \leq i < j \leq n$ with probability p. The goal is to design a node-DP algorithm that takes as input a graph $G \sim G(n,p)$ and outputs an estimate $\hat{p}$ of the edge density parameter p.

Surprisingly, until an elegant recent work of Borgs *et al.* [BCSZ18], the optimal accuracy for estimating the parameter p in a $G(n,p)$ via node-DP algorithms was unknown. Although that work essentially resolved the optimal accuracy of node-DP algorithms, their construction is again based on generic Lipschitz extensions, and thus results in an exponential-time algorithm, and, in our opinion, gives little insight for how to construct an efficient estimator with similar accuracy.

The main contribution of this work is to give a simple, polynomial-time estimator for Erdős-Rényi graphs whose error very nearly matches that of Borgs *et al.*'s estimator, and indeed matches it in a wide range of parameters. We achieve this by giving a more general result, showing how to optimally estimate the edge-density of any graph whose degree distribution is concentrated in a small interval.

6.1.1 Background on Node-Private Algorithms for Erdős-Rényi Graphs

Without privacy, the optimal estimator is simply to output the *edge-density* $p_G = |E|/\binom{n}{2}$ of the realized graph $G \sim G(n,p)$, which guarantees

$$\mathop{\mathbb{E}}_{G}\left[(p - p_G)^2\right] = \frac{p(1-p)}{\binom{n}{2}}.$$

The simplest way to achieve ε-node-DP is to add zero-mean noise to the edge-density with standard-deviation calibrated to its *global-sensitivity*, which is the amount that changing the neighborhood of a single node in a graph can change its edge-density. The global sensitivity of p_G is $\Theta(1/n)$, and thus the resulting private algorithm $\mathcal{A}_{naïve}$ satisfies

$$\mathop{\mathbb{E}}_{G}\left[(p - \mathcal{A}_{naïve}(G))^2\right] = \Theta\left(\frac{1}{\varepsilon^2 n^2}\right)$$

Note that this error is *at least* on the same order as the non-private error, and can asymptotically dominate the non-private error.

Borgs *et al.* [BCSZ18] gave an improved ε-node-DP algorithm such that, when both p and ε are $\gtrsim \log(n)/n$,

$$\mathbb{E}\left[(p - \mathcal{A}_{bcsz}(G))^2\right] = \underbrace{\frac{p(1-p)}{\binom{n}{2}}}_{\text{non-private error}} + \underbrace{\tilde{O}\left(\frac{p}{\varepsilon^2 n^3}\right)}_{\text{overhead due to privacy}}$$

What is remarkable about their algorithm is that, unless ε is quite small (roughly $\varepsilon \lesssim n^{-1/2}$), the first term dominates the error, in which case privacy comes essentially *for free.* That is, the error of the private algorithm is only larger than that of the optimal non-private algorithm by a $1 + o(1)$ factor. However, as we discussed above, this algorithm is not computationally efficient.

The only computationally efficient node-DP algorithms for computing the edge-density apply to graphs with small maximum degree [BBDS13, KNRS13, RS16], and thus do not give optimal estimators for Erdős-Rényi graphs unless p is very small.

6.1.2 Our Results

Our main result is a computationally efficient estimator for Erdős-Rényi graphs.

Theorem 6.1.1 (Erdős-Rényi Graphs, Informal). *There is an $O(n^2)$-time ε-node-DP algorithm $\mathcal{A}$ such that for every n and every $p \gtrsim 1/n$ if $G \sim G(n,p)$ then*

$$\mathop{\mathbb{E}}_{G,\mathcal{A}}\left[(p - \mathcal{A}(G))^2\right] = \underbrace{\frac{p(1-p)}{\binom{n}{2}}}_{\textit{non-private error}} + \underbrace{\tilde{O}\left(\frac{p}{\varepsilon^2 n^3} + \frac{1}{\varepsilon^4 n^4}\right)}_{\textit{overhead due to privacy}}$$

The error of Theorem 6.1.1 matches that of the exponential-time estimator of Borgs *et al.* [BCSZ18] up to the additive $\tilde{O}(1/\varepsilon^4 n^4)$ term, which is often not the dominant term in the overall error. In particular, the error of our estimator is still within a 1+o(1) factor of the optimal non-private error unless ε or p is quite small—for example, when p is a constant and $\varepsilon \gtrsim n^{-1/2}$.

Our estimator actually approximates the edge density for a significantly more general class of graphs than merely Erdős-Rényi graphs. Specifically, Theorem 6.1.1 follows from a more general result for the family of *concentrated-degree graphs.* For $k \in \mathbb{N}$, define $\mathcal{G}_{n,k}$ to be the set of n-node graphs such that the degree of every node is between $\bar{d} - k$ and $\bar{d} + k$, where $\bar{d} = 2|E|/n$ is the average degree of the graph.

Theorem 6.1.2 (Concentrated-Degree Graphs, Informal). *For every $k \in \mathbb{N}$, there is an $O(n^2)$-time ε-node-DP algorithm $\mathcal{A}$ such that for every n and every $G \in \mathcal{G}_{n,k}$,*

$$\mathop{\mathbb{E}}_{\mathcal{A}}\left[(p_G - \mathcal{A}(G))^2\right] = O\left(\frac{k^2}{\varepsilon^2 n^4} + \frac{1}{\varepsilon^4 n^4}\right)$$

where $p_G = |E|/\binom{n}{2}$ is the empirical edge density of G.

Theorem 6.1.1 follows from Theorem 6.1.2 by using the fact that for an Erdős-Rényi graph, with overwhelming probability the degree of every node lies in an interval of width $\tilde{O}(\sqrt{pn})$ around the average degree.

The main technical ingredient in Theorem 6.1.2 is to construct a *low sensitivity* estimator $f(G)$ for the number of edges. The first property we need is that when G satisfies the concentrated degrees property, $f(G)$ equals the number of edges in G. The second property of the estimator we construct is that its *smooth sensitivity* [NRS07] is low on these graphs G. At a high level, the smooth sensitivity of f at a graph G is the most that changing the neighborhood of a small number of nodes in G can change the value of $f(G)$. Once we have this property, it is sufficient to add noise to $f(G)$ calibrated to its smooth sensitivity. We construct f by carefully reweighting edges that are incident on nodes that do not satisfy the concentrated-degree condition.

Finally, we are able to show that Theorem 6.1.2 is optimal for concentrated-degree graphs. In additional to being a natural class of graphs in its own right, this lower bound demonstrates that in order to improve Theorem 6.1.1 we will need techniques that are more specialized to Erdős-Rényi graphs.

Theorem 6.1.3 (Lower Bound, Informal). *For every n and k, and every ε-node-DP algorithm $\mathcal{A}$, there is some $G \in \mathcal{G}_{n,k}$ such that*

$$\mathop{\mathbb{E}}_{\mathcal{A}}\left[(p_G - \mathcal{A}(G))^2\right] = \Omega\left(\frac{k^2}{\varepsilon^2 n^4} + \frac{1}{\varepsilon^4 n^4}\right)$$

The same bound applies to (ε, δ)-node-DP algorithms with sufficiently small $\delta \lesssim \varepsilon$.

6.2 Preliminaries

Let $\mathcal{G}_n$ be the set of n-node graphs. We say that two graphs $G, G' \in \mathcal{G}_n$ are *node-adjacent*, denoted $G \sim G'$, if G' can be obtained by G modifying the neighborhood of a single node i. That is, there exists a single node i such that for every edge e in the symmetric difference of G and G', e is incident on i. As is standard in the literature on differential privacy, we treat n as a fixed quantity and define adjacency only for graphs with the same number of nodes. We could easily extend our definition of adjacency to include adding or deleting a single node itself.

Definition 6.2.1 (Differential Privacy [DMNS06]). *A randomized algorithm $\mathcal{A} \leftarrow \mathcal{G}_n \rightarrow \mathcal{R}$ is (ε, δ)-node-differentially private if for every $G \sim G' \in \mathcal{G}_n$ and every $R \subseteq \mathcal{R}$,*

$$\mathbb{P}[\mathcal{A}(G) \in R] \leq e^{\varepsilon} \cdot \mathbb{P}[\mathcal{A}(G') \in R] + \delta$$

If $\delta = 0$ we will simply say that $\mathcal{A}$ is ε-node-differentially private. As we only consider node differential privacy in this work, we will frequently simply say that $\mathcal{A}$ satisfies differential privacy.

The next lemma is the basic composition property of differential privacy.

Lemma 6.2.1 (Composition [DMNS06]). *If $\mathcal{A}_1, \mathcal{A}_2 \leftarrow \mathcal{G}_n \rightarrow \mathcal{R}$ are each (ε, δ)-node-differentially private algorithms, then the mechanism $\mathcal{A}(G) = (\mathcal{A}_1(G), \mathcal{A}_2(G))$ satisfies $(2\varepsilon, 2\delta)$-node-differential privacy. The same holds if $\mathcal{A}_2$ may depend on the output of $\mathcal{A}_1$.*

We say that two graphs G, G' are at *node distance* c if there exists a sequence of graphs

$$G = G_0 \sim G_1 \ldots G_{c-1} \sim \ldots G_c = G'$$

The standard *group privacy* property of differential privacy yields the following guarantees for graphs at node distance $c > 1$.

Lemma 6.2.2 (Group Privacy [DMNS06]). *If $\mathcal{A} \leftarrow \mathcal{G}_n \rightarrow \mathcal{R}$ is (ε, δ)-node-differentially-private and G, G' are at node-distance c then for every $R \subseteq \mathcal{R}$,*

$$\mathbb{P}[\mathcal{A}(G) \in R] \leq e^{c\varepsilon}\mathbb{P}[\mathcal{A}(G') \in R] + ce^{c\varepsilon}\delta$$

Sensitivity and Basic DP Mechanisms. The main differentially private primitive we will use is *smooth sensitivity* [NRS07]. Let $f \leftarrow \mathcal{G}_n \rightarrow \mathbb{R}$ be a real-valued function. For a graph $G \in \mathcal{G}_n$, we can define the *local sensitivity* of f at G to be

$$LS_f(G) = \max_{G':G' \sim G} |f(G) - f(G')|$$

and the *global sensitivity* of f to be

$$GS_f = \max_{G} LS_f(G) = \max_{G' \sim G} |f(G) - f(G')|$$

A basic result in differential privacy says that we can achieve privacy for any real-valued function f by adding noise calibrated to the global sensitivity of f.

Theorem 6.2.3 (DP via Global Sensitivity [DMNS06]). *Let $f : \mathcal{G}_n \to \mathbb{R}$ be any function. Then the algorithm*

$$\mathcal{A}(G) = f(G) + \frac{GS_f}{\varepsilon} \cdot Z,$$

where Z is sampled from a standard Laplace distribution, satisfies $(\varepsilon, 0)$-differential privacy.[1] *Moreover, this mechanism satisfies $\mathop{\mathbb{E}}_{\mathcal{A}}[(\mathcal{A}(G) - f(G))^2] = O(GS_f/\varepsilon)$, and for all $t > 0$ we have that*

$$\mathop{\mathbb{P}}_{\mathcal{A}}[|\mathcal{A}(G) - f(G)| \geq t \cdot GS_f/\varepsilon] \leq \exp(-t).$$

In many cases the global sensitivity of f is too high, and we want to use a more refined mechanism that adds instance-dependent noise that is more comparable to the local sensitivity. This can be achieved via the *smooth sensitivity* framework of Nissim *et al.* [NRS07].

Definition 6.2.2 (Smooth Upper Bound [NRS07]). *Let $f : \mathcal{G}_n \to \mathbb{R}$ be a real-valued function and $\beta > 0$ be a parameter. A function $S : \mathcal{G}_n \to \mathbb{R}$ is a β-*smooth upper bound *on LS_f if*

1. *for all $G \in \mathcal{G}_n$, $S(G) \geq LS_f(G)$, and*

2. *for all neighboring $G \sim G' \in \mathcal{G}_n$, $S(G) \leq e^{\beta} \cdot S(G')$.*

The key result in smooth sensitivity is that we can achieve differential privacy by adding noise to $f(G)$ proportional to any smooth upper bound $S(G)$.

[1]The standard Laplace distribution Z has $\mathbb{E}[Z] = 0, \mathbb{E}[Z^2] = 2$, and density $\mu(z) \propto e^{-|z|}$.

Theorem 6.2.4 (DP via Smooth Sensitivity [NRS07, BS19]). *Let $f : \mathcal{G}_n \to \mathbb{R}$ be any function and S be a β-smooth upper bound on the local sensitivity of f for any $\beta \leq \varepsilon$. Then the algorithm*

$$\mathcal{A}(G) = f(G) + \frac{S(G)}{\varepsilon} \cdot Z,$$

where Z is sampled from a Student's t-distribution with 3 degrees of freedom, satisfies $(O(\varepsilon), 0)$-differential privacy.[2] *Moreover, for any $G \in \mathcal{G}_n$, this algorithm satisfies $\mathbb{E}_{\mathcal{A}}[(\mathcal{A}(G) - f(G))^2] = O(S(G)^2/\varepsilon^2)$.*

6.3 An Estimator for Concentrated-Degree Graphs

In this section we describe and analyze a node-differentially-private estimator for the edge density of a concentrated-degree graph.

6.3.1 The Estimator

In order to describe the estimator we introduce some key notation. The input to the estimator is a graph $G = (V, E)$ and a parameter k^*. Intuitively, k^* should be an upper bound on the concentration parameter of the graph, although we obtain more general results when k^* is not an upper bound, in case the user does not have an *a priori* upper bound on this quantity.

For a graph $G = (V, E)$, let $p_G = |E|/\binom{n}{2}$ be the empirical edge density of G, and let $\bar{d}_G = (n-1)p_G$ be the empirical average degree of G. Let k_G be the smallest positive integer value such that at most k_G vertices of G have degrees differing from $\bar{d}_G$ by more than $k' := k^* + 3k_G$. Define $I_G = [\bar{d}_G - k', \bar{d}_G + k']$. For each vertex $v \in V$, let $t_v = \min\{|t| : \deg_G(v) \pm t \in I_G\}$ be the distance between $\deg_G(v)$ and the interval I_G, and define the *weight*

[2]The Student's t-distribution with 3 degrees of freedom can be efficiently sampled by choosing $X, Y_1, Y_2, Y_3 \sim \mathcal{N}(0,1)$ independently from a standard normal and returning $Z = X/\sqrt{Y_1^2 + Y_2^2 + Y_3^2}$. This distribution has $\mathbb{E}[Z] = 0$ and $\mathbb{E}[Z^2] = 3$, and its density is $\mu(z) \propto 1/(1+z^2)^2$.

$\mathsf{wt}_G(v)$ of v as follows. For a parameter $\beta > 0$ to be specified later, let

$$\mathsf{wt}_G(v) = \begin{cases} 1 & \text{if } t_v = 0 \\ 1 - \beta t_v & \text{if } t_v \in (0, 1/\beta] \\ 0 & \text{otherwise.} \end{cases}$$

That is, $\mathsf{wt}_G(v) = \max(0, 1 - \beta t_v)$. For each pair of vertices $e = \{u, v\}$, define the *weight* $\mathsf{wt}_G(e)$ and *value* $\mathsf{val}_G(e)$ as follows. Let

$$\mathsf{wt}_G(e) = \min(\mathsf{wt}_G(u), \mathsf{wt}_G(v))$$

and let

$$\mathsf{val}_G(e) = \mathsf{wt}_G(e) \cdot x_e + (1 - \mathsf{wt}_G(e)) \cdot p_G$$

where x_e denotes the indicator variable on whether $e \in E$. As above, define the function f to be the total value of all pairs of vertices in the graph,

$$f(G) = \sum_{u,v \in V} \mathsf{val}_G(\{u, v\}),$$

where the sum is over unordered pairs of distinct vertices.

Once we construct this function f, we add noise to f proportional to a β-smooth upper bound on the sensitivity of f, which we derive in this section. Pseudocode for our estimator is given in Algorithm 11.

6.3.2 Analysis using Smooth Sensitivity

We begin by bounding the local sensitivity $LS_f(G)$ of the function f defined above.

Lemma 6.3.1. $LS_f(G) = O((k_G + k^*)(1 + \beta k_G) + \frac{1}{\beta})$.

Proof. Consider any pair of graphs G, G' differing in only a single vertex v^*, and note that the empirical edge densities p_G and $p_{G'}$ can differ by at most $\frac{2}{n} < \frac{2}{n-1}$, so $\bar{d}_G$ and $\bar{d}_{G'}$ can

Algorithm 11 Estimating the edge density of a concentrated-degree graph.

```
1: Input: A graph $G \in \mathcal{G}_n$ and parameters $\varepsilon > 0$ and $k^* \geq 0$.
2: Output: A parameter $0 \leq \hat{p} \leq 1$.
3: Let $p_G = \frac{1}{\binom{n}{2}} \sum_e x_e$ and $\bar{d}_G = (n-1)p_G$.
4: Let $\beta = \min(\varepsilon, 1/\sqrt{k^*})$.
5: Let $k_G > 0$ be the smallest positive integer such that at most $k_G$ vertices have degree
   outside $[\bar{d}_G - k^* - 3k_G, \bar{d}_G + k^* + 3k_G]$.
6: For $v \in V$, let $t_v = \min\{|t| : \deg_G(v) \pm t \in [\bar{d}_G - k^* - 3k_G, \bar{d}_G + k^* + 3k_G]\}$ and let
   $\mathsf{wt}_G(v) = \max(0, 1 - \beta t_v)$.
7: For each $u, v \in V$, let $\mathsf{wt}_G(\{u,v\}) = \min(\mathsf{wt}_G(u), \mathsf{wt}_G(v))$ and let $\mathsf{val}_G(e) = \mathsf{wt}_G(e) \cdot x_e +$
   $(1 - \mathsf{wt}_G(e))p_G$.
8: Let $f(G) = \sum_{u \neq v} \mathsf{val}_G(\{u,v\})$, where the sum is over unordered pairs of vertices.
9: Let $s = \max_{\ell \geq 0} ce^{-\beta\ell} \cdot (k_G + \ell + k^* + \beta(k_G + \ell)(k_G + \ell + k^*) + 1/\beta)$, where $c$ is the constant
   implied by Lemma 6.3.1.
10: Return $\frac{1}{\binom{n}{2}} \cdot (f(G) + (s/\varepsilon) \cdot Z)$, where $Z$ is sampled from a Student's $t$-distribution with
    three degrees of freedom.
```

differ by at most 2. Moreover, for any vertex $v \neq v^*$, the degree of v can differ by at most 1 between G and G'. Consequently, by the Triangle Inequality, for any $v \neq v^*$, $|\bar{d}_G - \deg_G(v)|$ can differ from $|\bar{d}_{G'} - \deg_{G'}(v)|$ by at most 3 and $\mathsf{wt}_G(v)$ can differ from $\mathsf{wt}_{G'}(v)$ by at most 3β. It follows from the former statement that k_G and $k_{G'}$ differ by at most 1.

Let Far_G denote the set of at most k_G vertices whose degree differs from $\bar{d}_G$ by more than $k' = k^* + 3k_G$. For any vertices $u, v \notin \mathsf{Far}_G \cup \mathsf{Far}_{G'} \cup \{v^*\}$, we have that $\mathsf{wt}_G(\{u,v\}) = \mathsf{wt}_{G'}(\{u,v\}) = 1$, and so $\mathsf{val}_G(\{u,v\}) = \mathsf{val}_{G'}(\{u,v\})$, since the edge $\{u,v\}$ is present in G if and only if it is present in G'.

Now consider edges $\{u,v\}$ such that $u, v \neq v^*$ but $u \in \mathsf{Far}_G \cup \mathsf{Far}_{G'}$ (and v may or may not be as well). If $\deg_G(u) \notin [\bar{d}_G - k'', \bar{d}_G + k'']$ for $k'' = k' + 1/\beta + 3$, then $\mathsf{wt}_G(u) = \mathsf{wt}_{G'}(u) = 0$ and so $|\mathsf{val}_G(\{u,v\}) - \mathsf{val}_{G'}(\{u,v\})| = |p_G - p_{G'}| \leq 2/n$. Otherwise, $\deg_G(u) \in [\bar{d}_G - k'', \bar{d}_G + k'']$. We can break up the sum

$$f_u(G) := \sum_{v \neq u} \mathsf{val}_G(\{u,v\}) = \sum_{v \neq u} \mathsf{wt}_G(\{u,v\}) \cdot x_{\{u,v\}} + \sum_{v \neq u} (1 - \mathsf{wt}_G(\{u,v\}))p_G.$$

Since at most k_G other vertices can have weight less than the weight of u, we can bound the first term by

$$\sum_{v \neq u} \mathsf{wt}_G(u) x_{\{u,v\}} \pm k_G \mathsf{wt}_G(u) = \deg_G(u)\mathsf{wt}_G(u) \pm k_G \mathsf{wt}_G(u)$$

and the second term by

$$p_G \cdot \left((n-1) - \sum_{v \neq u} \mathsf{wt}_G(\{u,v\}) \right) = \bar{d}_G - \bar{d}_G \mathsf{wt}_G(u) \pm p_G k_G \mathsf{wt}_G(u).$$

so the total sum is bounded by

$$f_u(G) = \bar{d}_G + (\deg_G(u) - \bar{d}_G)\mathsf{wt}_G(u) \pm 2k_G \mathsf{wt}_G(u).$$

Since $|\mathsf{wt}_G(u) - \mathsf{wt}_{G'}(u)| \leq 3\beta$, it follows that

$$|f_u(G) - f_u(G')| \leq 7 + 3\beta(k'' + 3) + 9\beta + 6\beta k_G = O(1 + \beta(k_G + k^*)).$$

Since there are at most $k_G + k'_G \leq 2k_G + 1$ vertices in $u \in \mathsf{Far}_G \cup \mathsf{Far}_{G'} \setminus \{v^*\}$, the total difference in the terms of $f(G)$ and $f(G')$ corresponding to such vertices is at most $O(k_G + \beta k_G(k_G + k^*))$. However, we are double-counting any edges between two vertices in $u \in \mathsf{Far}_G \cup \mathsf{Far}_{G'}$; the number of such edges is $O(k_G^2)$, and for any such edge e, $|\mathsf{val}_G(e) - \mathsf{val}_{G'}(e)| \pm O(\beta)$. Consequently the error induced by this double-counting is at most $O(\beta k_G^2)$, so the total difference between the terms of $f(G)$ and $f(G')$ corresponding to such vertices is still $O(k_G + \beta k_G(k_G + k^*))$.

Finally, consider the edges $\{u, v^*\}$ involving vertex v^*. If $\mathsf{wt}_G(v^*) = 0$ then

$$f_{v^*}(G) = \sum_{v \neq v^*} \mathsf{val}_G(\{v^*, v\}) = (n-1)p_G = \bar{d}_G.$$

If $\mathsf{wt}_G(v^*) = 1$ then $\deg_G(v^*) \in [\bar{d}_G - k', \bar{d}_G + k']$, so

$$\begin{aligned} f_{v^*}(G) &= \sum_{v \neq v^*} \mathsf{val}_G(\{v^*, v\}) \\ &= \deg_G(v^*) \pm k_G \\ &= \bar{d}_G \pm k' \pm k_G. \end{aligned}$$

Otherwise, $\deg_G(v^*) \in [\bar{d}_G - k' - 1/\beta, \bar{d}_G + k' + 1/\beta]$. Then we have that

$$\begin{aligned} f_{v^*}(G) &= \sum_{v \neq v^*} \mathsf{val}_G(\{v^*, v\}) \\ &= \bar{d}_G + (\deg_G(v^*) - \bar{d}_G)\mathsf{wt}_G(v^*) \pm k_G \mathsf{wt}_G(v^*) \\ &= \bar{d}_G \pm (\deg_G(v^*) - \bar{d}_G) \pm k_G, \end{aligned}$$

so in either case we have that $f_{v^*}(G) \in [\bar{d}_G - O(k_G + k^* + 1/\beta), \bar{d}_G + O(k_G + k^* + 1/\beta)]$. Consequently $|f_{v^*}(G) - f_{v^*}(G')| \leq O(k_G + k^* + 1/\beta)$.

Putting everything together, we have that $LS_f(G) = O((k_G + k^*)(1 + \beta k_G) + 1/\beta)$. □

We now compute a smooth upper bound on $LS_f(G)$. From the proof of Lemma 6.3.1, we have that there exists some constant $C > 0$ such that $LS_f(G) \leq C((k_G + k^*)(1 + \beta k_G) + \frac{1}{\beta})$. Let

$$g(k_G, k^*, \beta) = C((k_G + k^*)(1 + \beta k_G) + \tfrac{1}{\beta})$$

be this upper bound on $LS_f(G)$, and let

$$S(G) = \max_{\ell \geq 0} e^{-\ell\beta} g(k_G + \ell, k^*, \beta).$$

Lemma 6.3.2. *$S(G)$ is a β-smooth upper bound on the local sensitivity of f. Moreover,*

$$S(G) = O((k_G + k^*)(1 + \beta k_G) + \tfrac{1}{\beta}).$$

Proof. For neighboring graphs G, G', we have that

$$\begin{aligned}
S(G') &= \max_{\ell \geq 0} e^{-\ell\beta} g(k_{G'} + \ell, k^*, \beta) \\
&\leq \max_{\ell \geq 0} e^{-\ell\beta} g(k_G + \ell + 1, k^*, \beta) \\
&= e^{\beta} \max_{\ell \geq 1} e^{-\ell\beta} g(k_G + \ell, k^*, \beta) \\
&\leq e^{\beta} \max_{\ell \geq 0} e^{-\ell\beta} g(k_G + \ell, k^*, \beta) \\
&= e^{\beta} S(G).
\end{aligned}$$

Moreover, for fixed k_G, k^*, β, consider the function $h(\ell) = e^{-\ell\beta} g(k_G + \ell, k^* \beta)$, and consider the derivative $h'(\ell)$. We have that

$$h'(\ell) = C\beta e^{-\ell\beta}(k_G + \ell)(1 - \beta(k_G + \ell + k^*)).$$

Consequently the only possible local maximum for $\ell > 0$ would occur for $\ell = 1/\beta - k_G - k^*$; note that the function h decreases as $\ell \to \infty$. Consequently the maximum value of h occurs for some $\ell \leq 1/\beta$, and so

$$\begin{aligned}
S(G) &= \max_{\ell \geq 0} h(\ell) \\
&= \max_{\ell \geq 0} ce^{-\ell\beta}(k_G + \ell + k^* + (k_G + \ell)(k_G + \ell + k^*)\beta + 1/\beta) \\
&\leq C \cdot (k_G + 1/\beta + k^* + (k_G + 1/\beta)(k_G + 1/\beta + k^*)\beta + 1/\beta) \\
&= C \cdot (3k_G + 2k^* + \beta k_G(k_G + k^*) + 3/\beta) \\
&= O((k_G + k^*)(1 + \beta k_G) + 1/\beta)
\end{aligned}$$

as desired. □

Theorem 6.3.3. *Algorithm 11 is $(O(\varepsilon), 0)$-differentially private. Moreover, for any k-concentrated n-vertex graph $G = (V, E) \in \mathcal{G}_{n,k}$ with $k \geq 1$, we have that Algorithm 11* *satisfies*

$$\mathop{\mathbb{E}}_{\mathcal{A}}\left[\left(\frac{|E|}{\binom{n}{2}} - \mathcal{A}_{\varepsilon,k}(G)\right)^2\right] = O\left(\frac{k^2}{\varepsilon^2 n^4} + \frac{1}{\varepsilon^4 n^4}\right)$$

Proof. Algorithm 11 computes function f and releases it with noise proportional to a β-smooth upper bound on the local sensitivity for $\beta \leq \varepsilon$. Consequently $(O(\varepsilon), 0)$-differential privacy follows immediately from Theorem 6.2.4.

We now analyze its accuracy on k-concentrated graphs G. If G is k-concentrated and $k^* \geq k$, then $\mathsf{wt}_G(v) = 1$ for all vertices $v \in V$ and $\mathsf{val}_G(\{u,v\}) = x_{\{u,v\}}$ for all $u, v \in V$, and so $f(G) = |E|$. Consequently Algorithm 11 computes the edge density of a k-concentrated graph with noise distributed according to the Student's t-distribution scaled by a factor of $S(G)/(\varepsilon\binom{n}{2})$.

Since G is k-concentrated, we also have that $k_G = 1$, and so $S(G) = O(k + \beta(k+1) + 1/\beta) \leq O(k + 1/\varepsilon)$ by Lemma 6.3.2. The variance of the Student's t-distribution with three degrees of freedom is $O(1)$, so the expected squared error of the algorithm is

$$O\left(\frac{(k+1/\varepsilon)^2}{\varepsilon^2 n^4}\right) = O\left(\frac{k^2}{\varepsilon^2 n^4} + \frac{1}{\varepsilon^4 n^4}\right)$$

as desired. □

6.4 Application to Erdős-Rényi Graphs

In this section we show how to apply Algorithm 11 to estimate the parameter of an Erdős-Rényi graph. Pseudocode is given in Algorithm 12.

Algorithm 12 Estimating the parameter of an Erdős-Rényi graph.

```
1: Input: A graph $G \in \mathcal{G}_n$ and parameters $\varepsilon, \alpha > 0$.
2: Output: A parameter $0 \le \hat{p} \le 1$.


3: Let $\tilde{p}' \leftarrow \frac{1}{\binom{n}{2}} \sum_e x_e + (2/\varepsilon n) \cdot Z$, where $Z$ is a standard Laplace
4: Let $\tilde{p} \leftarrow \tilde{p}' + 4\log(1/\alpha)/\varepsilon n$ and $\tilde{k} \leftarrow \sqrt{\tilde{p}n}\log(n/\alpha)$
5: Return $\hat{p} \leftarrow \mathcal{A}_{\tilde{k},\varepsilon}(G)$ where $\mathcal{A}_{\tilde{k},\varepsilon}$ is Algorithm 11 with parameters $\tilde{k}$ and $\varepsilon$
```

It is straightforward to prove that this mechanism satisfies differential privacy.

Theorem 6.4.1. *Algorithm 12 satisfies $(O(\varepsilon), 0)$-node-differential privacy.*

Proof. The first line computes the empirical edge density of the graph G, which is a function with global sensitivity $(n-1)/\binom{n}{2} = 2/n$. Therefore by Theorem 6.2.3 this step satisfies $(\varepsilon, 0)$-differential privacy. The third line runs an algorithm that satisfies $(O(\varepsilon), 0)$-differential privacy for every fixed parameter $\tilde{k}$. By Lemma 6.2.1, the composition satisfies $(O(\varepsilon), 0)$-differential privacy. $\square$

Next, we argue that this algorithm satisfies the desired accuracy guarantee.

Theorem 6.4.2. *For every $n \in \mathbb{N}$ and $\frac{1}{2} \ge p \ge 0$, and an appropriate parameter $\alpha > 0$, Algorithm 12 satisfies*

$$\mathop{\mathbb{E}}_{G \sim G(n,p), \mathcal{A}} \left[(p - \mathcal{A}(G))^2\right] = \frac{p(1-p)}{\binom{n}{2}} + \tilde{O}\left(\frac{\max\{p, \frac{1}{n}\}}{\varepsilon^2 n^3} + \frac{1}{\varepsilon^4 n^4}\right)$$

Proof. We will prove the result in the case where $p \ge \frac{\log n}{n}$. The case where p is smaller will follow immediately by using $\frac{\log n}{n}$ as an upper bound on p. The first term in the bound is simply the variance of the empirical edge-density $\bar{p}$. For the remainder of the proof we will focus on bounding $\mathbb{E}\left[(\bar{p} - \hat{p})^2\right]$.

A basic fact about $G(n,p)$ for $p \ge \frac{\log n}{n}$ is that with probability at least $1 - 2\alpha$: (1) $|\bar{p} - p| \le 2\log(1/\alpha)/n$, and (2) the degree of every node i lies in the interval $[\bar{d} \pm \sqrt{pn}\log(n/\alpha)]$ where $\bar{d}$ is the average degree of G. We will assume for the remainder that these events hold.

Using Theorem 6.2.3, we also have that with probability at least $1-\alpha$, the estimate $\tilde{p}'$ satisfies $|\bar{p}-\tilde{p}'| \leq 4\log(1/\alpha)/\varepsilon n$. We will also assume for the remainder that this latter event holds. Therefore, we have $p \leq \tilde{p}$ and $p \geq \tilde{p} - 8\log(1/\alpha)/\varepsilon n$.

Assuming this condition holds, the graph will have $\tilde{k}$-concentrated degrees for $\tilde{k}$ as specified on line 2 of the algorithm. Since this assumption holds, we have

$$\begin{aligned}
\mathbb{E}\left[(\bar{p}-\mathcal{A}_{\tilde{k},\varepsilon}(G))^2\right] &= \tilde{O}\left(\frac{\tilde{k}^2}{\varepsilon^2 n^4}+\frac{1}{\varepsilon^4 n^4}\right)\\
&= \tilde{O}\left(\frac{\tilde{p}n}{\varepsilon^2 n^4}+\frac{1}{\varepsilon^4 n^4}\right)\\
&= \tilde{O}\left(\frac{pn+\frac{1}{\varepsilon n}}{\varepsilon^2 n^4}+\frac{1}{\varepsilon^4 n^4}\right)\\
&= \tilde{O}\left(\frac{pn}{\varepsilon^2 n^4}+\frac{1}{\varepsilon^4 n^4}\right).
\end{aligned}$$

To complete the proof, we can plug in a suitably small $\alpha = 1/\mathrm{poly}(n)$ so that the $O(\alpha)$ probability of failure will not affect the overall mean-squared error in a significant way. $\square$

6.5 Lower Bounds for Concentrated-Degree Graphs

In this section we prove a lower bound for estimating the number of edges in concentrated-degree graphs. Theorem 6.1.3, which lower bounds the mean squared error follows by applying Jensen's Inequality.

Theorem 6.5.1. *For every $n,k \in \mathbb{N}$, every $\varepsilon \in [\frac{2}{n},\frac{1}{4}]$ and $\delta \leq \frac{\varepsilon}{32}$, and every (ε,δ)-node-DP algorithm A, there exists $G \in \mathcal{G}_{n,k}$ such that $\mathbb{E}_{A}[|p_G - A(G)|] = \Omega\left(\frac{k}{\varepsilon n^2}+\frac{1}{\varepsilon^2 n^2}\right)$.*

The proof relies on the following standard fact about differentially private algorithms. Since we are not aware of a formal treatment in the literature, we include a proof for completeness.

Lemma 6.5.2. *Suppose there are two graphs $G_0, G_1 \in \mathcal{G}_{n,k}$ at node distance at most $\frac{1}{\varepsilon}$ from* *one another. Then for every $(\varepsilon, \frac{\varepsilon}{32})$-node-DP algorithm A, there exists $b \in \{0,1\}$ such that*

$$\mathop{\mathbb{E}}_{A}\left[|p_{G_b} - A(G_b)|\right] = \Omega\left(|p_{G_0} - p_{G_1}|\right).$$

Proof. Let A be any ε-node-DP algorithm. Since G_0, G_1 have node distance at most $\frac{1}{\varepsilon}$, by group privacy (Lemma 6.2.2), for every set S and every $b \in \{0,1\}$

$$\mathbb{P}\left[A(G_b) \in S\right] \leq e \cdot \mathbb{P}\left[A(G_{1-b}) \in S\right] + \tfrac{1}{16}.$$

Now, let $S_b = \left\{y : |y - p_{G_b}| < \frac{1}{2}\left|p_{G_0} - p_{G_1}\right|\right\}$ and note that S_0 and S_1 are disjoint by construction. Let $\rho = \min\{\mathbb{P}\left[A(G_0) \in S_0\right], \mathbb{P}\left[A(G_1) \in S_1\right]\}$. Then we have

$$\begin{aligned}
1 - \rho &\geq \mathbb{P}\left[A(G_0) \notin S_0\right] \\
&\geq \mathbb{P}\left[A(G_0) \in S_1\right] \\
&\geq e^{-1}\mathbb{P}\left[A(G_1) \in S_1\right] - \tfrac{1}{16} \\
&\geq e^{-1}\rho - \tfrac{1}{16}
\end{aligned}$$

from which we can deduce $\rho \leq \frac{4}{5}$. Therefore, for some $b \in \{0,1\}$, we have

$$\mathbb{P}\left[|p_{G_b} - A(G_b)| \geq \tfrac{1}{2}\left|p_{G_0} - p_{G_1}\right|\right] \geq \tfrac{1}{5},$$

from which the lemma follows. □

We will construct two simple pairs of graphs to which we can apply Lemma 6.5.2.

Lemma 6.5.3 (Lower bound for large k). *For every $n, k \in \mathbb{N}$ and $\varepsilon \geq 2/n$, there is a pair of graphs $G_0, G_1 \in \mathcal{G}_{n,k}$ at node distance $1/\varepsilon$ such that $|p_{G_0} - p_{G_1}| = \Omega(\frac{k}{\varepsilon n^2})$.*

Proof. Let G_0 be the empty graph on n nodes. Note that $p_{G_0} = 0$, $\bar{d}_{G_0} = 0$, and G_0 is in $\mathcal{G}_{n,k}$.

We construct G_1 as follows. Start with the empty bipartite graph with $\frac{1}{\varepsilon}$ nodes on the left and $n - \frac{1}{\varepsilon}$ nodes on the right. We connect the first node on the left to each of the first

k nodes on the right, then the second node on the left to each of the next k nodes on the right and so on, wrapping around to the first node on the right when we run out of nodes. By construction, $p_{G_1} = k/\varepsilon\binom{n}{2}$, $\bar{d}_{G_1} = 2k/\varepsilon n$. Moreover, each of the first $\frac{1}{\varepsilon}$ nodes has degree exactly k and each of the nodes on the right has degree

$$\frac{k/\varepsilon}{n - 1/\varepsilon} \pm 1 = \frac{k}{\varepsilon n - 1} \pm 1$$

Thus, for n larger than some absolute constant, every degree lies in the interval $[\bar{d}_{G_1} \pm k]$ so we have $G_1 \in \mathcal{G}_{n,k}$. □

Lemma 6.5.4 (Lower bound for small k). *For every $n \geq 4$ and $\varepsilon \in [2/n, 1/4]$, there is a pair of graphs $G_0, G_1 \in \mathcal{G}_{n,1}$ at node distance $1/\varepsilon$ such that $|p_{G_0} - p_{G_1}| = \Omega(\frac{1}{\varepsilon^2 n^2})$.*

Proof. Let $i = \lceil n\varepsilon \rceil$, and let G_0 be the graph consisting of i disjoint cliques each of size $\lfloor n/i \rfloor$ or $\lceil n/i \rceil$. Let G_1 be the graph consisting of $i+1$ disjoint cliques each of size $\lfloor n/(i+1) \rfloor$ or $\lceil n/(i+1) \rceil$. We can obtain G_0 from G_1 by taking one of the cliques and redistributing its vertices among the i remaining cliques, so G_0 and G_1 have node distance $\ell := \lfloor n/(i+1) \rfloor \leq 1/\varepsilon$. For $1/4 \geq \varepsilon \geq 2/n$ we have that $\ell \geq \lfloor 1/2\varepsilon \rfloor > 1/4\varepsilon$. Transforming G_1 into G_0 involves removing a clique of size ℓ, containing $\binom{\ell}{2}$ edges, and then inserting these ℓ vertices into cliques that already have size ℓ, adding at least ℓ^2 new edges. Consequently G_0 contains at least $\ell^2 - \ell(\ell-1)/2 = \ell(\ell+1)/2$ more edges than G_1, so

$$|p_{G_1} - p_{G_0}| \geq \frac{\binom{\ell+1}{2}}{\binom{n}{2}} \geq \frac{\ell^2}{n^2} \geq \Omega(1/\varepsilon^2 n^2),$$

as desired. □

Theorem 6.5.1 now follows by combining Lemmas 6.5.2, 6.5.3, and 6.5.4.

Bibliography

[AAD$^+$04] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC*, 2004.

[AAD$^+$06] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

[AAE07] Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. In *DISC*, 2007.

[AAE$^+$17] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *SODA*, pages 2560–2579, 2017.

[AAER07] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[AAFJ08] Dana Angluin, James Aspnes, Michael J. Fischer, and Hong Jiang. Self-stabilizing population protocols. *TAAS*, 3(4):13:1–13:28, 2008.

[AAG18] Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *SODA*, 2018.

[ABBS16] James Aspnes, Joffroy Beauquier, Janna Burman, and Devan Sohier. Time and space optimal counting in population protocols. In *OPODIS*, 2016.

[AIKW13] B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In *Crypto*, 2013.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, 1999.

[BBDS12] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *FOCS*, 2012.

[BBDS13] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, 2013.

[BCC$^+$15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *ESORICS*, 2015.

[BCD$^+$09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *FC*, 2009.

[BCG04] E. R Berlekamp, John Horton Conway, and R. K. Guy. Winning ways for your mathematical plays. In *A K Peters 1st Ed.*, 2001-2004.

[BCP15] E. Boyle, K-M. Chung, and R. Pass. Large-scale secure computation: Multi-party computation for (parallel) ram programs. In *Crypto*, 2015.

[BCSZ18] Christian Borgs, Jennifer T. Chayes, Adam D. Smith, and Ilias Zadik. Revealing network structure, confidentially: Improved rates for node-private graphon estimation. In *59th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '18, pages 533–543, Paris, France, 2018.

[BDOZ11] Rikke Bendlin, Ivan Damgard, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Eurocrypt*, 2011.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In *Crypto*, 1995.

[Bea96] D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *STOC*, 1996.

[BGT13] E. Boyle, S. Goldwasser, and S. Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In *TCC*, 2013.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.

[BH08] Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure MPC with linear communication complexity. In *TCC*, 2008.

[BHKR13] M. Bellare, V.T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE Security and Privacy*, 2013.

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In *TCC*, 2017.

[BK10] Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. *IACR Cryptology ePrint Archive 2010/086*, 2010.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptology*, 22(1):114–138, 2009.

[BLO18] Carsten Baum, Huang Lin, and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. Cryptology ePrint Archive 2018/107, 2018.

[BLW08] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*, 2008.

[BMM99] Amos Beimel, Tal Malkin, and Silvio Micali. The all-or-nothing nature of two-party secure computation. In *Crypto*, 1999.

[BNSV15] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially private release and learning of threshold functions. *arXiv preprint arXiv:1504.07553*, 2015.

[BOO10] Amos Beimel, Eran Omri, and Ilan Orlov. Protocols for multiparty coin toss with dishonest majority. In *Crypto*, 2010.

[BS19] Mark Bun and Thomas Steinke. Smooth sensitivity, revisited. Manuscript, 2019.

[BSFO12] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Crypto*, 2012.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, 1988.

[CD18] Rachel Cummings and David Durfee. Individual sensitivity preprocessing for data privacy. *arXiv preprint arXiv:1804.08645*, 2018.

[CDD$^+$01] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *Eurocrypt*, 2001.

[CDLN14] Alejandro Cornejo, Anna R. Dornhaus, Nancy A. Lynch, and Radhika Nagpal. Task allocation in ant colonies. In *DISC*, 2014.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *CRYPTO*, 1997.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *Eurocrypt*, 2010.

[CK89] B. Chor and E. Kushilevitz. A zero-one law for boolean privacy (extended abstract). In *STOC*, 1989.

[CKM$^+$19] Clément L. Canonne, Gautam Kamath, Audra McMillan, Jonathan Ullman, and Lydia Zakynthinou. Differentially private identity testing for multivariate distributions. Manuscript, 2019.

[CN82] GJ Chang and GL Nemhauser. The k-domination and k-stability problem on graphs. *Techn. Report*, 540, 1982.

[CO15] T. Chou and C. Orlandi. The simplest protocol for oblivious transfer. In *Latincrypt*, 2015.

[Coo04] Matthew Cook. Universality in elementary cellular automata, 2004.

[CPP18] Ran Canetti, Sunoo Park, and Oxana Poburinnaya. Fully bideniable interactive encryption. *IACR Cryptology ePrint Archive*, 2018:1244, 2018.

[CSS10] TH Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *Automata, Languages and Programming*, pages 405–417. Springer, 2010.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *Eurocrypt*, 1991.

[DDWY90] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. In *FOCS*, 1990.

[DFGR06] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Eric Ruppert. When birds die: Making population protocols fault-tolerant. In *DCOSS*, 2006.

[DH97] Shlomi Dolev and Ted Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago J. Theor. Comput. Sci.*, 1997.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *Crypto*, 2006.

[Dij74] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communcations of the ACM*, pages 643–644, 1974.

[DIK$^+$08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam D. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *Crypto*, 2008.

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Eurocrypt*, 2010.

[DKM$^+$06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Eurocrypt*, 2006.

[DKMS12] V. Dani, V. King, M. Movahedi, and J. Saia. Brief announcement: breaking the o(nm) bit barrier, secure multiparty computation with a static adversary. In *PODC*, 2012.

[DKS99] Ivan Damgård, Joe Kilian, and Louis Salvail. On the (im) possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Eurocrypt*, 1999.

[DLECW92] G Martinez De La Escalera, AL Choi, and Richard I Weiner. Generation and synchronization of gonadotropin-releasing hormone (gnrh) pulses: intrinsic properties of the gt1-1 gnrh neuronal cell line. *Proceedings of the National Academy of Sciences*, 89(5):1852–1855, 1992.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[DMPTH10] Tal Danino, Octavio Mondragón-Palomino, Lev Tsimring, and Jeff Hasty. A synchronized quorum of genetic clocks. *Nature*, 463(7279):326, 2010.

[DN07a] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Crypto*, 2007.

[DN07b] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6), 2007.

[DNPR10] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *STOC*, 2010.

[DPSZ12] I. Damgard, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Crypto*, 2012.

[DR13] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2013.

[DRV10] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *FOCS*, 2010.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *Crypto*, 1982.

[FFGS07] M. Fitzi, M. Franklin, J.A. Garay, and H.V. Simhadri. Towards optimal and efficient perfectly secure message transmission. In *TCC*, 2007.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *PKC*, 2007.

[Gar70] Martin Gardner. The fantastic combinations of john conway's new solitaire game "life", 1970.

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In *TCC*, 2017.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[GKKO07] Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *FOCS*, 2007.

[GKP$^+$13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.

[GLM$^+$10] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *SODA*, 2010.

[GMRL15] Mohsen Ghaffari, Cameron Musco, Tsvetomira Radeva, and Nancy A. Lynch. Distributed house-hunting in ant colonies. In *PODC*, 2015.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In *Crypto*, 1992.

[GOSS18] Shafi Goldwasser, Rafail Ostrovsky, Alessandra Scafuro, and Adam Sealfon. Population stability: Regulating size in the presence of an adversary. In *PODC*, 2018.

[GPV07] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. *IACR Cryptology ePrint Archive 2007/432*, 2007.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[GR17] Oded Goldreich and Dana Ron. On learning and testing dynamic environments. *Journal of the ACM (JACM)*, 64(3):21, 2017.

[GRU12] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *TCC*, 2012.

[GV87] Oded Goldreich and Ronen Vainish. How to solve any protocol problem - an efficiency improvement. In *Crypto*, 1987.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *TCC*, 2008.

[HHR$^+$14] Justin Hsu, Zhiyi Huang, Aaron Roth, Tim Roughgarden, and Zhiwei Steven Wu. Private matchings and allocations. In *STOC*, 2014.

[HIK07] Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Crypto*, 2007.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. Ot-combiners via secure computation. In *TCC*, 2008.

[HKE13] Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *Crypto*, 2013.

[HKK$^+$14] Y. Huang, J. Katz, V. Kolesnikov, R. Kumaresan, and A. Malozemoff. Amortizing garbled circuits. In *Crypto*, 2014.

[HKN$^+$05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Eurocrypt*, 2005.

[HLM13] M. Hirt, C. Lucas, and U. Maurer. A dynamic tradeoff between active and passive corruptions in secure multi-party computation. In *Crypto*, 2013.

[HLMJ09] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, 2009.

[IEH$^+$16] Ai Ishida, Keita Emura, Goichiro Hanaoka, Yusuke Sakai, and Keisuke Tanaka. Group signature with deniability: How to disavow a signature. In *CANS*, 2016.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Crypto*, 2006.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Crypto*, 2003.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Crypto*, 2008.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, 1989.

[JMPT87] HJ Jongsma, M Masson-Pevet, and L Tsjernina. The development of beat-rate synchronization of rat myocyte pairs in cell culture. *Basic research in cardiology*, 82(5):454–464, 1987.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, 1988.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014.

[KNRS13] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476. Springer, 2013.

[KRS16] Ranjit Kumaresan, Srinivasan Raghuraman, and Adam Sealfon. Network oblivious transfer. In *Crypto*, 2016.

[KRSY14] Vishesh Karwa, Sofya Raskhodnikova, Adam D. Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *ACM Transactions on Database Systems*, 39(3):22:1–22:33, 2014.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP*, 2008.

[KS16] Vishesh Karwa and Aleksandra Slavković. Inference using noisy degrees: Differentially private β-model and synthetic graphs. *Annals of Statistics*, 44(1):87–112, 2016.

[KST54] Tamás Kovári, Vera Sós, and Pál Turán. On a problem of k. zarankiewicz. In *Colloquium Mathematicum*, volume 1:3, pages 50–57, 1954.

[Kus89] E. Kushilevitz. Privacy and communication complexity. In *FOCS*, 1989.

[LBBC14a] Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Conscious and unconscious counting on anonymous dynamic networks. In *ICDCN*, 2014.

[LBBC14b] Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Counting in anonymous dynamic networks under worst-case adversary. In *ICDCS*, 2014.

[Lin13] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In *Crypto*, 2013.

[LNWX17] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS*, 2017.

[LOS14] E. Larraia, E. Orsini, and N.P. Smart. Dishonest majority multi-party computation for binary circuits. In *Crypto*, 2014.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Eurocrypt*, 2007.

[LPSY15] Y. Lindell, B. Pinkas, N.P. Smart, and A. Yanai. Efficient constant round multi-party computation combining bmr and spdz. In *Crypto*, 2015.

[LR14] Y. Lindell and B. Riva. Cut-and-choose yao-based two-party computation with low cost in the online/offline and batch settings. In *Crypto*, 2014.

[LSW06] Joseph K. Liu, Willy Susilo, and Duncan S. Wong. Ring signature with designated linkability. In *IWSEC*, 2006.

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP*, 2004.

[Man00] R. Mankiewicz. *The story of mathematics*. The story of mathematics. Princeton University Press, 2000.

[MBB$^+$13] Carlos Aguilar Melchor, Slim Bettaieb, Xavier Boyen, Laurent Fousse, and Philippe Gaborit. Adapting lyubashevsky's signature schemes to the ring signature setting. In *Africacrypt*, 2013.

[MM75] A Meir and JW Moon. Relations between packing and covering numbers of a tree. *Pacific J. Math*, 61(1):225–233, 1975.

[MO09] Joseph S Markson and Erin K O'Shea. The molecular clockwork of a protein-based circadian oscillator. *FEBS letters*, 583(24):3938–3947, 2009.

[Mon] Monero. Monero: Private digital currency.

[Mor78] Robert Morris. Counting large numbers of events in small registers. In *Communications of the ACM*, pages 840–842, 1978.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Eurocrypt*, 2012.

[MPR10] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. A zero-one law for cryptographic complexity with respect to computational UC security. In *Crypto*, 2010.

[MPW07] Remo Meier, Bartosz Przydatek, and Jürg Wullschleger. Robuster combiners for oblivious transfer. In *TCC*, 2007.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[MR13] P. Mohassel and B. Riva. Garbled circuits checking garbled circuits: More efficient and secure two-party computation. In *Crypto*, 2013.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, 1999.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology - CT-RSA*, 2005.

[NNOB12] J. Nielsen, P. Nordholt, C. Orlandi, and S. Burra. A new approach to practical active-secure two-party computation. In *Crypto*, 2012.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, 2001.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.

[PS19] Sunoo Park and Adam Sealfon. It wasn't me! repudiability and unclaimability of ring signatures. In *Crypto*, 2019.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Crypto*, 2008.

[Rab81] M. O. Rabin. How to exchange secrets with oblivious transfer. In *Technical Report TR-81, Aiken Computation Lab, Harvard University*, 1981.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, 1989.

[RS16] Sofya Raskhodnikova and Adam D. Smith. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *57th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '16, pages 495–504, New Brunswick, NJ, USA, 2016.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Asiacrypt*, 2001.

[Sea16] Adam Sealfon. Shortest paths and distances with differential privacy. In *PODS*, 2016.

[SS10] Sven Schäge and Jörg Schwenk. A cdh-based ring signature scheme with short signatures and public keys. In *FC*, 2010.

[SU19] Adam Sealfon and Jonathan Ullman. Efficiently estimating erdos-renyi graphs with node differential privacy. *CoRR*, abs/1905.10477, 2019.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

[vN51] John von Neumann. The general and logical theory of automata, 1951.

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[Wul07] Jürg Wullschleger. Oblivious-transfer amplification. In *Eurocrypt*, 2007.

[WW06] Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In *Eurocrypt*, 2006.

[XCT14] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *20th ACM International Conference on Knowledge Discovery and Data Mining*, KDD'14, pages 911–920, 2014.

[XY04] Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *CARDIS*, pages 271–286, 2004.

[Yao86] Andrew Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, 1986.

[ZMS14] M. Zamani, M. Movahedi, and J. Saia. Millions of millionaires: Multiparty computation in large networks. In *ePrint 2014/149*, 2014.

[ZRE15] S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *Eurocrypt*, 2015.