Cryptographic Error Correction

by

Christopher Jason Peikert

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
[September 2006]
July 2006

Author .....
Department of Electrical Engineering and Computer Science
July 31, 2006

Certified by .....
Silvio Micali
Professor of Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Cryptographic Error Correction

by

Christopher Jason Peikert

Submitted to the Department of Electrical Engineering and Computer Science
on July 31, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

It has been said that "cryptography is about concealing information, and coding theory is about revealing it." Despite these apparently conflicting goals, the two fields have common origins and many interesting relationships.

In this thesis, we establish new connections between cryptography and coding theory in two ways: first, by applying cryptographic tools to solve classical problems from the theory of error correction; and second, by studying special kinds of codes that are motivated by cryptographic applications.

In the first part of this thesis, we consider a model of error correction in which the source of errors is *adversarial*, but limited to *feasible* computation. In this model, we construct appealingly simple, general, and efficient cryptographic coding schemes which can recover from much larger error rates than schemes for classical models of adversarial noise.

In the second part, we study *collusion-secure fingerprinting codes*, which are of fundamental importance in cryptographic applications like data watermarking and traitor tracing. We demonstrate *tight* lower bounds on the lengths of such codes by devising and analyzing a general collusive attack that works for any code.

Thesis Supervisor: Silvio Micali
Title: Professor of Computer Science

Acknowledgments

In writing acknowledgments, my greatest fear is that I will fail to adequately convey my thanks to, and the full contributions of, everyone who has helped me through my schooling and education (not always the same things). With that caveat, I will do my best.

My advisor, Silvio Micali, has navigated me through the Labyrinth that is graduate school. What Minotaurs I did encounter always arose despite his advice, never because of it. I thank him for excellent instruction, intriguing research problems, rewarding teaching opportunities, high expectations, steadfast patience, personal warmth, and great humor.

The other members of my thesis committee, Ron Rivest and Shafi Goldwasser, have been equally wonderful. I have had the great fortune of taking courses from, teaching with, and receiving expert research advice from both of them over several years. I look back very fondly on all these experiences; their value is incalculable.

I also thank Madhu Sudan for both his academic guidance and a lot of fun times. His excellent teaching first sparked my interest in coding theory, and his clear insights into even my most nebulous questions have inspired a large portion of my graduate research.

With their eagerness to provide expert assistance in all matters of laboratory survival (including, most importantly, proper nutrition!), Be "Mom" Blackburn and Joanne Talbot Hanley somehow manage to keep the whole place up and running, and make it look easy in the process.

I also heartily thank all my collaborators, office mates, fellow students, and combinations thereof (also known as "friends"): Ben Adida, Adi Akavia, Simson Garfinkel, Seth Gilbert, Jonathan Herzog, Susan Hohenberger, Yael Tauman Kalai, Adam Klivans, Swastik Kopparty, Matt Lepinski, David Liben-Nowell, Moses Liskov, Anna Lysyanskaya, Alantha Newman, Ryan O'Donnell, Rafael Pass, Sofya Raskhodnikova, Leo Reyzin, Alon Rosen, Guy Rothblum, abhi shelat, Adam Smith, Vinod Vaikuntanathan, Paul Valiant, Grant Wang, Steve Weis, David Woodruff, Sergey Yekhanin, and Hanson Zhou. I could not have enjoyed better company!

Finally, and most of all, I thank all the members of my family for their overwhelming love and support, which no words can accurately describe.

Contents

1 Introduction **11**
- 1.1 A Brief History of Cryptography and Coding . . . 11
- 1.2 Our Contributions . . . 15
  - 1.2.1 Coding for Feasible Channels . . . 16
  - 1.2.2 Lower Bounds for Fingerprinting Codes . . . 16

2 Preliminaries **19**
- 2.1 Notation . . . 19
- 2.2 Coding Theory Background . . . 19
- 2.3 Cryptography Background . . . 20

3 Error Correction for Feasible Adversarial Channels **23**
- 3.1 Channels for Communication . . . 23
- 3.2 Contributions . . . 25
  - 3.2.1 Overview of the Construction . . . 26
- 3.3 Comparison with Related Work . . . 27
  - 3.3.1 List Decoding . . . 27
  - 3.3.2 List Decoding with Side Information . . . 27
  - 3.3.3 Code Scrambling . . . 28
  - 3.3.4 Private Codes . . . 31
  - 3.3.5 Multicast Authentication . . . 32
- 3.4 Formal Model and the Coding Game . . . 32
- 3.5 Constructing Robust Coding Schemes . . . 37
  - 3.5.1 Intuition . . . 37
  - 3.5.2 Formal Construction . . . 38
  - 3.5.3 Proof of Robustness . . . 39
  - 3.5.4 Concrete Instantiations . . . 43
  - 3.5.5 Optimism and Optimizations . . . 44
- 3.6 Optimality and Necessity of the Model . . . 46
  - 3.6.1 Necessity of Local State . . . 47
  - 3.6.2 Necessity of Sender Secrecy . . . 48
  - 3.6.3 Necessity of a Bounded Channel . . . 49
- 3.7 Symmetric-Key Model and Construction . . . 50
  - 3.7.1 Model and Coding Game . . . 51
  - 3.7.2 Construction and Security Reduction . . . 51

4 Lower Bounds for Fingerprinting Codes **53**
4.1 Fingerprinting and Watermarking 53
4.1.1 Introducing Coalitions 54
4.1.2 Our Contributions 55
4.1.3 Prior Work 55
4.2 The Boneh-Shaw Model 56
4.3 Our Lower Bound 57
4.3.1 Overview of the Argument 58
4.3.2 Min-Mass of Distributions 59
4.3.3 Relating Min-Mass to Tracing Error 59
4.3.4 The Coalitions and Their Strategies 60
4.3.5 Min-Mass of Binomial Distributions 63
4.3.6 Proof of Theorem 65

List of Figures

3-1 Formal definition of a channel interacting with a coding scheme in the coding game, and the notion of a successful channel. 35
3-2 Pictorial representation of the coding game. Time flows downward. . 35
3-3 Pictorial representation of the sender and receiver in the coding scheme. 40
3-4 Definition of the forger algorithm $\mathcal{F}_{k'}$. 41

4-1 Graph of the bias function $b(\cdot)$ 61

Chapter 1

Introduction

Cryptography and coding theory have had rich, intertwined histories. The mathematical foundations of both fields were first established in the 1940s — just one year apart — by the same founder, Claude E. Shannon [61, 62]. Even today, the two fields seem to remain inseparable: the phrases "cryptography" and "coding theory" appear together in the titles of countless books, conferences, scholarly papers, lectures, research groups, and computer science courses.

Coding theory is primarily the study of two main problems, called *source coding* and *channel coding*. The former is concerned with representing information from a data source as compactly as possible. While a fascinating problem, it will not be our area of interest in this thesis. Instead, we will focus much of our attention on the latter problem, which is concerned with communicating *reliably* over an *unreliable* (or *noisy*) *channel*.

In the theory of cryptography, the two oldest and most essential problems are *encryption* and *message authentication*. The former is concerned with *concealing* communication in the presence of an *eavesdropper*, while the latter deals with *confirming the provenance* of a communication in the presence of a potential *imitator*.

While all of the above are disparate goals, their basic settings are all quite similar: in every case, a source of messages, or *sender*, is attempting to communicate with a *receiver*, with some kind of antagonistic entity between them. These similarities are fundamental, and account for many of the interrelations between cryptography and coding to date.

1.1 A Brief History of Cryptography and Coding

The road to modern cryptography. Shannon was the first the give cryptography a formal, mathematical treatment [62], focusing primarily on the task of encryption. In his model of a cryptosystem (or as he called it, a "secrecy system"), Shannon defined secrecy in terms of the amount of *information* a ciphertext (i.e., an encrypted message) reveals about its underlying message. To obtain *perfect secrecy*, a cryptosystem would need to reveal *zero* information about an encrypted message. That is, the *a posteriori* probability distribution over the space of possible messages (given

the ciphertext, but not the key) would have to be equal to the *a priori* distribution. A fundamental result is that it is impossible to obtain perfect secrecy unless both communicating parties share a perfectly uniform, secret (to the eavesdropper) string having length at least that of the message being communicated. In fact, the classical Vernam cipher (also known as the one-time pad) meets this requirement exactly, and hence one might consider the one-time pad an "optimal" solution, and the problem of encryption "closed." Unfortunately, proper use of a one-time pad is extremely impractical and inconvenient, due to the large amount of key material that must be generated, managed, and securely distributed (the most difficult aspect of implementing the scheme, by far).

The explosion of progress in cryptography in modern times has been achieved not by finding better ways of fulfilling Shannon's definition (which would be mathematically impossible), but by proposing more realistic notions of eavesdroppers (and adversaries in general) and more useful definitions of security. Because Shannon's perfect secrecy is defined in terms of *information*, there is an implicit assumption that the eavesdropper can perform an unbounded amount of computation. For example, in Shannon's formulation, a cryptosystem must hide messages even from an eavesdropper that enumerates all messages and, by brute-force, calculates the *a posteriori* probability of each message given the transmitted ciphertext. Depending on the method of encryption, such an eavesdropper may be completely infeasible in the real world, but Shannon's definition requires that it be thwarted.

The growth in the field of computational complexity in the 1970s provided a catalyst for more realistic notion of a *computationally feasible* cryptographic adversary, and indeed the entirely new paradigm of *public-key cryptography*, first proposed in the open literature by Diffie and Hellman [20]. With the discovery of the RSA function [58], the basic tools and assumptions of complexity-based, number-theoretic cryptography were in place. However, the field still lacked a formal *definition* of secure encryption for a feasible adversary akin to Shannon's perfect secrecy, and without such a definition it was impossible to objectively analyze the security of a proposed method of encryption. The work of Goldwasser and Micali [32] filled this gap with an elegant notion called *semantic security*, and provided a cryptosystem that was semantically secure under a plausible complexity assumption. The essence of semantic security is that no *feasible eavesdropper*, given the encryption of a message, can guess *any property* of the message with a probability significantly better than if it had not seen the ciphertext at all! In other words, semantic security guarantees that encryption "hides all properties" of the underlying message (almost) perfectly, from the point of view of a feasible adversary. The similarity to Shannon's perfect secrecy is easy to see: while perfect security *hides all information* about an encrypted message (even to an unbounded eavesdropper), semantic security *hides all properties* of the message from a computationally feasible adversary.

Since the early 1980s, complexity-based cryptography has proliferated, with the development of formal notions for numerous security-related notions, such as pseudorandomness [10, 71, 29], authentication (including identification schemes [26], message authentication codes, and signatures [34]), commitment [9], zero-knowledge [33], stronger forms of encryption [53, 22, 5, 18], multiparty computation [31], and more.

The road to modern coding theory. A central problem of coding theory is reliable communication over an unreliable channel. All solutions to this problem, in some form or another, rely on the basic idea of encoding messages with some *redundancy*, allowing the receiver to detect and correct whatever errors may arise during transmission through the channel. The main goal is to minimize the amount of redundancy while maximizing the quantity of errors that can be corrected. There are, however, two main approaches to formalizing this basic idea, whose essential differences are in how the channel is modelled.

The first formal model is due to Shannon [61], who modelled the channel as a *probabilistic* (random) process. Specifically, the channel introduces errors according to a *fixed, known* probability distribution that is oblivious to the transmitted message. In an elegant use of the probabilistic method [3], Shannon showed that there exist asymptotically good encodings which can correct errors with arbitrarily good probability — in fact, a random encoding is almost a good one! However, there are some drawbacks to Shannon's approach: first, the result is only *asymptotic*, applying only for sufficiently long blocks of data. In addition, the result is only *existential* (not explicit), so it does not say *how* to encode or correct errors, only that it is possible to do so. Furthermore, the basic model of the channel may not be entirely satisfactory: while it may be reasonable to assume that the noise distribution is fixed and oblivious to the message, assuming that the distribution is *known* ahead of time to the code designer is only reasonable in the most controlled environments.

Two years after Shannon's work, Hamming [41] took a more conservative approach, modelling the channel as an *adversary* with a limited allowance of errors. In Hamming's formulation, not only may the channel's behavior be unknown to the code designer, but its errors may also *depend on the message*. Furthermore, the channel may also draw upon *unlimited computational resources*. This is certainly a strong model for a channel, and the *error-correcting codes* designed for this regime are unquestionably robust. However, the strength of the channel also incurs some downsides; for example, it is impossible to unambiguously decode (in the worst case) when the number of errors exceeds half the length of the data block.

Today, nearly all the work on coding theory in computer science is focused on two main aspects of Hamming's theory of error-correcting codes: combinatorial properties, and efficient algorithms. In the first direction, progress has come from using algebraic and combinatorial techniques to improve the important properties of codes (e.g., distance, information rate, alphabet size, list-decoding radius). In the algorithmic direction, there has been rapid progress in devising efficient encoding and (list-)decoding algorithms for many different families of codes [66, 63, 39, 40, 37]. However, the bulk of the attention paid to computational complexity has been focused on the *encoding* and *decoding* algorithms; the complexity of the *channel* is rarely (if ever) discussed.

The role of coding in cryptography. Cryptography has frequently been a "consumer" of coding theory's concepts and tools, especially in recent years. As a first example, consider *secret-sharing schemes*, which were first proposed by Shamir [60], and

which have since become essential to threshold cryptography, secure multiparty computation, and many other essential areas of cryptography. A secret-sharing scheme is a method of distributing a secret piece of information among several players so that (1) no small group of players can learn anything about the secret, and (2) the players can come together and reliably reconstruct the secret, even if some of them are faulty. This latter property appears suspiciously similar to the goal of coding theory, and in fact, most secret-sharing schemes are based directly upon error-correcting codes. Shamir's original scheme, for example, corresponds directly to the Reed-Solomon family of codes [57, 50].

Another way in which coding theory has contributed to progress in cryptography is via a notion called *list decoding*, which was first proposed (as a combinatorial notion) independently by Elias and Wozencraft [25, 70], and which was first shown to be algorithmically feasible by Sudan [66]. This notion is useful in contexts where the amount of error in a channel is so large that it is impossible to *uniquely* identify the intended message. In such a setting, list decoding provides a small set of *all possible* candidate messages. This idea first proved useful in cryptography with Goldreich and Levin's construction of a universal hard-core predicate for any one-way function [30]. Specifically, Goldreich and Levin found an efficient list-decoding algorithm for the *Hadamard code*, which corresponds naturally to their proposed hard-core predicate. While this "list-decoding perspective" on their work has only become clear in retrospect [67], it has proven essential in providing a unifying framework for existing hard-core predicates [10, 2, 43] and demonstrating new hard predicates [1]. List decoding has also proven useful in many other cryptographic contexts, including zero-knowledge protocols [24], traitor tracing [64], and quantum hard-core functions [44].

The role of cryptography in coding. While cryptography and coding have matured concurrently, the relationship between them has historically been very one-sided: as explained above, cryptography (and complexity theory in general) has often been a "consumer" of ideas and results from coding. However, it has rarely reciprocated as a "provider." That is, the rich set of tools and notions that have been developed in cryptography have not, to date, been used to solve any problems in coding. Nor have cryptographic problems provided much motivation for new notions and goals in coding theory. However, there are a few notable exceptions, two of which we discuss below.

In 1994, Lipton proposed an intriguing application of the modern cryptographic methodology to information theory and coding [47]. Noting that Hamming's version of a channel is adversarial and computationally *unbounded*, Lipton instead proposed a more realistic channel which is adversarial, but like most cryptographic adversaries, is limited to *feasible computation*. While Lipton articulated this core idea over a decade ago, it was not known until recently whether a feasible channel could be exploited to any benefit.

Certain applications in cryptography have also motivated entirely new notions of error-correcting codes and noisy channels. Specifically, there are many information systems in which users have access to data that they are not authorized to redis-

tribute. Examples of such data include: commercial software, classified or confidential documents, cryptographic keys stored in specialized devices for decoding encrypted broadcasts (e.g., cable TV decoders, DVD players, satellite receivers, etc.), and copyrighted audio/video content. Because information is inherently copyable, it is very difficult to design technical measures that *prohibit* redistribution of data. Instead, there have been several proposals to *deter* redistribution by robustly embedding some uniquely-identifying information (called "marks") into each user's copy. The rationale is that a user incriminates himself by exposing his copy widely enough for an authority to acquire it and extract its marks. However, there is an additional complication: a set users may *collude* by creating a functionally useful "hybrid" from pieces of their different copies. Methods for reliably ascertaining at least one of the colluders from such a hybrid copy go under the general rubric of *traitor tracing* schemes [16].

There is a natural way of viewing these schemes from the perspective of coding theory: the sequences of marks embedded in each copy make up the *codewords* of some code. The colluders, in piecing together their individual copies, act as a certain kind of *noisy channel* whose input is a set of codewords, and whose output is the hybrid copy (which must resemble the input in some defined way). The authority then attempts to *decode* the hybrid back to one of the original codewords. From this viewpoint, the goal of traitor tracing is to design specialized codes and efficient decoding algorithms for this unique noise model. Such codes go by several different names (such as *collusion-secure fingerprinting codes* [13], *frameproof* and *traceability codes* [65], and *IPP-* and *TA-codes* [64], among others) depending on the details of the model, but their essential goals are the same.

1.2 Our Contributions

In this thesis, we aim to make the relationship between cryptography and coding theory more symbiotic. We do this in two ways: first, we employ a cryptographic perspective in modelling and solving a fundamental problem in coding. Second, we use cryptographic applications as motivation for studying new kinds of codes.

Our first contribution is a cryptographic approach to reliable communication over a noisy channel. We propose a strong, novel model of a channel that is limited to feasible computation. For such a channel, we then construct coding schemes that tolerate *significantly* more errors than schemes for classical channels; in fact, the number of errors tolerated is *optimal* for *any* reasonable channel model.

In the second part, we turn our attention to *collusion-secure fingerprinting codes*, which are motivated by cryptographic applications like watermarking and traitor tracing. We focus on the lengths of such codes as a function of the number of potential colluders. By devising and analyzing a probabilistic collusion strategy, we demonstrate a lower bound on the code length. This bound is tight, as it matches (asymptotically) a subsequent construction by Tardos [68].

1.2.1 Coding for Feasible Channels

Chapter 3 contains our cryptographic approach to reliable communication over noisy channels. This portion of the thesis first appeared as a preliminary conference version in a paper by Micali, Peikert, Sudan, and Wilson [51].

We put forward the first formal model for communication over an adversarial channel that is limited to feasible computation. Our model precisely specifies the setup assumptions for a coding scheme, the capabilities and goals of the channel, and the requirements for robustness.

Our setup and operational assumptions are relatively mild (indeed, they are weaker than the assumptions in all related work; see Section 3.3): we assume that the sender generates a pair of keys, one of which it keeps secret from the channel, and the other which it publishes in some way that allows the receiver to obtain a one-time, error-free copy. We also assume that the sender can maintain a small amount of local (non-secret) state, such as a clock or nonce.

Our model for the channel is quite strong: the channel has adaptive control over all the messages that are encoded by the sender, it has access to all the information that is available to the receiver, and it can employ arbitrary probabilistic polynomial-time computation. In addition to a large allowance of noise, it also has the power to "drop" arbitrary messages (i.e., refuse to deliver them at all), without even alerting the receiver to the fact that messages have been dropped.

In the face of such a channel, our definition of a robust coding scheme is one that allows the receiver to efficiently, correctly, and unambiguously recover every (non-dropped) message with only a vanishingly small probability of error. We construct such a coding scheme (actually, a broad class of schemes) via an elegant combination of two powerful tools: *authentication* (from cryptography) and *list decoding* (from coding theory).

Our scheme yields great benefits in error correction for both binary and large alphabets. Namely,

1. *For binary alphabets, we construct coding schemes having positive information rate which are uniquely decodable from a* $1/2 - \gamma$ *error rate for any constant* $\gamma > 0$.

2. *For large alphabets, we construct coding schemes which are uniquely decodable from any error rate up to* $1 - R$ *for any information rate* $R > 0$.

In both cases, the error rate tolerated by our scheme is optimal.

In addition, we show that every major property of our model is *essential* for decoding beyond the classical barriers imposed by a Hamming channel. Thus our construction can be seen as the "best possible" use of the bounded-channel assumption for error correction.

1.2.2 Lower Bounds for Fingerprinting Codes

Chapter 4 is concerned with special kind of codes, called *collusion-secure fingerprinting codes* [13], that are motivated by the traitor-tracing problem. This portion of

the thesis first appeared as a preliminary conference version in a paper by Peikert, Shelat, and Smith [55].

One of the most important properties of a fingerprinting code is its length. Depending on the application (e.g., data watermarking, broadcast encryption), the length of the code may dictate the number of marks that must be robustly embedded into a work, or the amount of computation that must be done to decrypt a ciphertext. Therefore minimizing the length of fingerprinting codes is one of the highest priorities for a practical implementation.

We investigate bounds on the lengths of fingerprinting codes, as a function of the number of colluders that they are designed to defeat. Our analysis implies that, in order to be secure, the codes must have length that is essentially *quadratic* in the number of potential colluders.

In order to prove this bound, we devise a general collusive strategy that works for any fingerprinting code. Our strategy is probabilistic and simple to describe: the coalition employs independent *biased coin flips* at each position of the code, where the bias is given by a carefully-chosen function of the symbols in the colluders' codewords.

The analysis of the strategy is more complicated. First, we define a novel statistical measure, which we call the *min-mass* of a set of distributions. We then relate min-mass to the error probability of any tracing algorithm for the fingerprinting code. Finally, we analyze the min-mass of our coalition strategies by expressing it in terms of the statistical distance between binomial distributions, which we compute via their information divergence. We conclude that when the fingerprinting code is too short, the min-mass for our collusive strategy is large, and the code is insecure. The lower bound follows by the contrapositive.

Chapter 2

Preliminaries

In this chapter we introduce some of the notation that we will use throughout the thesis, and briefly recall some concepts and definitions from coding theory and cryptography that we will rely upon later.

2.1 Notation

For a positive integer n, we write $[n] = \{1, \ldots, n\}$.

For a finite set Σ and a nonnegative integer n, we will refer to an element of Σ^n as a *string* or *word* over the *alphabet* Σ. For a word $x \in \Sigma^n$, we write x_i to denote the ith component of x.

Asymptotics. For positive functions $f(n), g(n)$, we write $f(n) = O(g(n))$ and/or $g(n) = \Omega(f(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C$ for some absolute constant $C \geq 0$. We write $f(n) = o(g(n))$ and/or $g(n) = \omega(f(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

We say that a function $f(n)$ is *negligible* in n if, for any positive polynomial $p(n)$, $f(n) = o(1/p(n))$.

2.2 Coding Theory Background

Basic concepts. An *error-correcting code* over an alphabet Σ is a mapping $C : \Sigma^k \to \Sigma^n$ from strings of length k to strings of length $n \geq k$; k is called the *message length* and n is called the *block length*. The *(information) rate* of the code is $\frac{k}{n}$. The set $\{C(x) : x \in \Sigma^k\}$ is also called the *code*, and its elements are called *codewords*. We will often overload notation and write C to refer to the set of codewords (as well as the mapping).

The *Hamming distance* between two words $x, y \in \Sigma^n$ is the number of positions i in which x_i and y_i differ: $\Delta(x, y) = |\{i \in [n] : x_i \neq y_i\}|$. It is easy to verify that the Hamming distance is a metric. A code C has *distance* (at least) d if $\Delta(C(x), C(y)) \geq d$ for every distinct $x, y \in \Sigma^k$.

We say that a code C is (e, L)-*list decodable* if, for every $r \in \Sigma^n$, there are at most L codewords within distance e of r: $|\{x \in C \text{ and } \Delta(x, r) \leq e\}| \leq L$.

Asymptotics and complexity. In this work we will be concerned both with the asymptotics of code parameters, and with the computational complexity of algorithms associated with codes. As a consequence, we consider infinite *families* of codes

$$\mathbb{C} = \left\{ C_k \ : \ \Sigma_k^k \rightarrow \Sigma_k^{n(k)} \right\}_{k \in T},$$

where $T \subseteq \mathbb{N}$ is an infinite set (usually, we will have $T = \mathbb{N}$). Some of the families we employ will use the full generality of the definition, having alphabets Σ_k depending on k; others will not, letting Σ_k be some fixed Σ for all k. The family has distance $d(k)$ if for every $k \in T$, C_k has distance $d(k)$. The aim of most error-correcting codes is to have $n(k)$ grow as slowly as possible, while having $d(k)$ grow as quickly as possible. The *relative rate* of the code family is $\liminf_{k \rightarrow \infty} \frac{k}{n(k)}$, and the *relative distance* is $\liminf_{k \rightarrow \infty} \frac{d(k)}{n(k)}$.

We say that a family of codes is *explicitly constructible* if it has a single efficient (deterministic polynomial-time) algorithm for computing the encoding functions C_k, and one for their inverses C_k^{-1} (for all $k \in T$). All the codes we employ will be explicitly constructible, so we will freely alternate between $x \in \Sigma_k^k$ and $C_k(x) \in \Sigma_k^{n(k)}$. We say that a family is *efficiently* $(e(k), L(k))$-*list decodable* if it has a single efficient algorithm $\mathcal{LD}$ which, for all $k \in T$, finds all the (at most) $L(k)$ codewords in C_k within distance $e(k)$ of any input string $r \in \Sigma_k^{n(k)}$.

2.3 Cryptography Background

Digital signatures. Digital signatures were conceived of as far back as Diffie and Hellman's seminal paper [20], but were not fully formalized and constructed until Goldwasser, Micali, and Rivest [34] introduced the notion of *existential unforgeability* under *chosen-message attack*. This notion states that an adversary should not be able to generate a valid signature for *any* message, even after seeing signatures for any other messages of its choice.

A digital signature scheme $\mathcal{SS}$ is a triple of poly-time randomized algorithms $\mathcal{SS} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ (respectively, the *key generation*, *signing*, and *verification* algorithms) having the following properties:

1. On input 1^k (where k is the *security parameter*), $\mathcal{G}$ outputs (pk, sk), where pk is the *public (verification) key* and sk is the *secret (signing) key*.

2. On the secret key sk and a message $m \in \{0,1\}^*$, $\mathcal{S}$ outputs a *signature* σ such that $\mathcal{V}_{pk}(m, \sigma) = \mathcal{V}(pk, m, \sigma) = 1$ with probability 1 (taken over the randomness of $\mathcal{G}$, $\mathcal{S}$, and $\mathcal{V}$).

   In addition, our applications will require the length of the signature to be fixed (for a given security parameter). Without loss of generality, we will assume that

the length $|\sigma|$ of the signature is k.

3. Even with adaptive oracle access to the signer, no efficient *forger* can produce a valid (m', σ') pair for an unqueried m', except with negligible probability. We make this formal below.

Let $\mathcal{F} = \{\mathcal{F}_k\}_{k\in\mathbb{N}}$ be a family of randomized oracle circuits. Consider the following *chosen-message attack*, parameterized by k:

1. On input 1^k, $\mathcal{G}$ outputs (pk, sk).

2. On input pk, $\mathcal{F}_k^{\mathcal{S}_{sk}}$ outputs a pair (m', σ'), where $\mathcal{S}_{sk}$ represents an oracle to the algorithm $\mathcal{S}(sk, \cdot)$, which $\mathcal{F}_k$ may query adaptively.

3. We say that $\mathcal{F}_k$ *succeeds* in the attack if $\mathcal{V}_{pk}(m', \sigma') = 1$ and m' was *not* one of the queries issued by $\mathcal{F}_k$ in the previous step.

Define

$$\mathsf{Adv\text{-}CMA}_{\mathcal{F}}^{\mathcal{SS}}(k) = \Pr[\mathcal{F}_k \text{ succeeds against } \mathcal{SS} \text{ for security parameter } k],$$

where the probability is taken over the random coins of $\mathcal{SS}$ and $\mathcal{F}$.

For functions $q(k)$, $s(k)$, define

$$\mathsf{Adv\text{-}CMA}_{q,s}^{\mathcal{SS}} = \max_{\mathcal{F}} \left\{ \mathsf{Adv\text{-}CMA}_{\mathcal{F}}^{\mathcal{SS}}(k) \right\},$$

where the maximum is taken over all $\mathcal{F} = \{\mathcal{F}_k\}$ such that $\mathcal{F}_k$ issues at most $q(k)$ queries in any execution of the chosen-message attack (for security parameter k), and has circuit size at most $s(k)$. We say that a signature scheme is *existentially unforgeable under chosen-message attack* if, for every $q(k), s(k) \in \mathrm{poly}(k)$, $\mathsf{Adv\text{-}CMA}_{q,s}^{\mathcal{SS}}$ is a negligible function in k.

Constructions and signature length. Secure signatures (according to the definition above) were first constructed by Goldwasser, Micali, and Rivest [34] under the assumption that there exist *claw-free pairs of permutations* (which follows from an assumption on either the hardness of factoring or of computing discrete logs).

In subsequent work, Bellare and Micali [4] demonstrated that *trapdoor permutations* were sufficient for obtaining secure signatures. Later, Naor and Yung [52] constructed secure signatures from *universal one-way hash functions*, which were shown by Rompel [59] to be equivalent to *one-way functions*.

Depending on the desired level of exact security, different schemes produce signatures of different lengths. For the concrete instantiations of our coding schemes, we will get better performance by employing signatures which are *as short as possible*. Boneh *et al* [12, 11] have constructed short, secure signatures from some relatively non-standard assumptions.

Message authentication codes (MACs). Message authentication codes are the symmetric-key analog of digital signatures. They are comprised of key generation, authentication, and verification algorithms. The main difference is that the key generator only outputs a single (secret) key sk, which is an input to *both* the authentication and verification algorithms, but which is not known to the forger. In the chosen-message attack, the forger is given access to oracles that compute the authentication and verification algorithms (using the secret key), and attempts to produce an authentic pair (m', σ') where m' was not a query to the authenticator oracle. Due to the similarity to digital signatures, and the ancillary uses of MACs in our applications, we omit a more formal definition.

Chapter 3

Error Correction for Feasible Adversarial Channels

The theory of error correction is concerned with sending information reliably over a "noisy channel" that introduces errors into the transmitted data. The goal of this theory is to design *coding schemes* which are capable of detecting and correcting such errors. The setting is usually modelled as follows: a *sender* starts with some *message*, which is represented as a string of symbols over some *alphabet*. The sender *encodes* the message into a longer string over the same alphabet, and transmits the block of data over a *channel*. The channel introduces *errors* (or *noise*) by changing some of the symbols of the transmitted block, then delivers the corrupted block to the *receiver*. Finally, the receiver attempts to *decodes* the block, hopefully to the intended message. Whenever the sender wants to transmit a new message, the process is repeated.

Two quantities are of special interest in this setting. The first is the *information rate*, which is the ratio of the message length to the encoded block length. This is a measure of how much "actual message data" is carried by each transmitted symbol. The second is the *error rate*, which is the ratio of the number of errors to the block length. This is a measure of how "noisy" the channel is, i.e. how much data it corrupts.

Of course, we desire coding schemes that tolerate high error rates while simultaneously having large information rates. In practice, smaller alphabets are desirable too, as most digital communication devices are, at their lowest levels, capable of interpreting only binary digits (bits).

3.1 Channels for Communication

There are two historically popular ways to model a noisy channel for communication. In his 1948 work that pioneered the field of information theory [61], Shannon considered a *randomized channel*, which perturbs each transmitted symbol independently with some fixed probability. Using a beautiful and elegant probabilistic argument, Shannon demonstrated how information can be encoded to withstand such noise with probability arbitrarily close to 1. Unfortunately, Shannon's notion of a probabilistic channel may be too simplistic for modelling actual sources of noise. In the real world,

data corruption can be the result of complex interactions among many factors both intrinsic and extrinsic to the communication system, and errors tend to be distributed neither uniformly nor independently. For example, both radio interference and physical defects on storage media tend to be "bursty," with periods of very high noise separated by spans of quiescence.

Partly in response to these issues, in his work two years later on *error-correcting codes* [41], Hamming proposed an *adversarial channel* that perturbs symbols in a *worst-case* fashion (subject only to an upper bound on the number of errors per block of data). This model of a channel is much more "pessimistic" (or, depending on one's outlook, "safer") than Shannon's, as it encompasses any arbitrarily-complex source of noise. As such, it gives much stronger guarantees on the robustness of the resulting coding scheme. Unfortunately, this robustness comes at a price: it imposes severe limits on the performance of codes. For instance, in order to correctly decode the intended message under a binary alphabet, the error rate of the channel must not exceed 1/4 (unless the block lengths grow exponentially with the message lengths, which is of no practical use).

Under Hamming's notion of a channel, Elias and Wozencraft independently proposed *list decoding* [25, 70] as a way to recover from higher error rates. The idea is to *relax* the task of decoder, allowing it to output a short *list* of all possible messages (though saying nothing about which is the correct one). For a worst-case adversarial channel with a high error rate, list decoding seems to be the best one can hope for — but under a more *"realistic"* model of an adversarial channel, is it possible to *uniquely* decode under high error rates?

The computational complexity of channels. While Hamming's conception of a channel is always described as *adversarial*, it is also — though this is rarely stated explicitly — *computationally unbounded*. That is, the channel may not only introduce errors in the worst possible locations, but it may also employ enormous computational resources to decide *where* those locations are.

In 1994, Lipton [47] put forward the notion of a *feasible* channel, which is essentially a Hamming channel restricted to *polynomial-time* computation. That is, the channel still introduces errors adversarially (always subject to a given error rate), but only by employing a tractable amount of computation.

Remarkably, under standard cryptographic assumptions and assuming that sender and receiver share secret randomness, Ding, Gopalan, and Lipton [21] proved that for such a bounded channel, it is possible to decode uniquely from high error rates. Their scheme requires the communicating parties to share a secret key which is unknown to the channel.

More significantly, though the bounded-channel model was first envisioned over a decade ago, nobody has yet shown an *essential* use of this assumption to yield any unique benefits over an unbounded channel. That is, previous constructions still work when the channel is computationally unbounded, as long as the sender and receiver share sufficient secret randomness. The bounded-channel assumption is used to reduce the *amount* of shared randomness that is needed, but not to eliminate

it altogether. This computational assumption is thus an *additional* one, and does not supplant the assumption of secret randomness shared between the sender and receiver.

3.2 Contributions

Our model and cryptographic setting. We propose the first precise model for communicating over a feasible adversarial channel. Our model describes precisely and explicitly the capabilities and goals of the channel, and rigorously defines the notion of a correct coding scheme for such a channel. Previous works treated these issues only implicitly, and in fact our channel has significantly stronger capabilities than the channels that have been considered in related work. Specifically, our channel knows everything that the receiver knows, and also has the power to "drop" messages at will, without even alerting the receiver to the fact that messages have been dropped.

The sender (but *not* the receiver) keeps a small amount of local state information, which he uses when encoding messages. This state can be realized by, for example, a local counter or a local clock. We stress that the clock need not be synchronized with any other clocks in the outside world, but that it only produce monotonically increasing values.

We work in a very simple cryptographic setting: we assume that one-way functions exist (the "minimal" cryptographic assumption) and that the sender has a public key known to the receiver (and, perhaps, known to the channel as well).

Our results. Our setting yields great benefits in error correction for both binary and large alphabets. Namely,

1. *For a binary alphabet, we construct positive-rate coding schemes which are uniquely decodable under error rate* $1/2 - \gamma$*, for any constant* $\gamma > 0$*.*

   Classically, a $1/4 - \gamma$ error rate is the best possible for unique decoding (and positive information rate). We stress that *in any reasonable channel model* (even Shannon's probabilistic one), decoding of *any kind* (even list decoding) is impossible under an error rate of $1/2$. Therefore this result is optimal in a very strong sense, and matches the best possible error rates in the weaker Shannon model.

2. *For large alphabets, we construct coding schemes which are uniquely decodable under error rate up to* $1 - R$ *for any information rate* $R > 0$*.*

   Under an error rate of $1 - R$, an information rate of R is called the *Shannon capacity* of the channel, and is again optimal for any alphabet size. This can be seen by a simple (informal) counting argument: in order to communicate $k = Rn$ symbols, at least $k = Rn$ symbols meaningful symbols must emerge through the channel, meaning at most $(1 - R)n$ symbols may be altered.

We point out one (almost paradoxical) consequence of this particular result: for small values of R, we can allow the channel to *adversarially* corrupt almost the *entire codeword*, and still recover the unique intended message.

To achieve these results, we actually prove a very general *reduction* from unique decoding against a feasible channel to list decoding against an arbitrary channel. Specifically,

> If one-way functions exist, *unique decoding* from e errors in the feasible-channel model reduces to *poly-time list decoding* from e errors in the Hamming model (using the same alphabet, with no asymptotic loss in information rate).

We obtain results 1 and 2 above by applying this reduction to the list-decodable binary codes of Guruswami and Sudan [40], and the *Folded Reed-Solomon* codes of Guruswami and Rudra [38], respectively.

Necessity of the model. There are three defining characteristics of our model: (1) the sender is stateful while the receiver is stateless, (2) the sender keeps a secret key which is unknown to the channel while the receiver keeps no secrets at all, and (3) the channel is assumed to be computationally bounded.

We show that every aspect of this model is essential for decoding beyond the classical barriers of the Hamming model. That is, relaxing any of these three requirements makes the task of unique decoding under high error rates *impossible*. Thus our construction can be seen as the "best possible" use of the bounded-channel assumption for error correction. See Section 3.6 for details.

3.2.1 Overview of the Construction

Starting with any list-decodable code, we define a large (but sparse) subset of *authentic* codewords, which make up an *authentic subcode*. Authentic codewords have the following essential properties: (1) they are easy for the sender to create, (2) they are hard for an adversary to produce (even after seeing other authentic codewords), and (3) they are easy for anyone, the receiver in particular, to recognize. We then specify a *cryptographic sieving* procedure, which specifies a way to select a *unique* authentic word from an arbitrary list of codewords (some authentic, some possibly not).

With the two notions of an *authentic subcode* and *cryptographic sieve*, we can sketch the method of communicating over a feasible channel:

- The sender, when encoding a message, always constructs an authentic codeword and sends it over the channel.
- The receiver, upon receiving a corrupted word, first *list decodes* it. Of course, list decoding only provides a set of candidate codewords. In order to uniquely decode, the recipient uses the cryptographic sieve to select the appropriate authentic codeword.

Provided that the number of errors is suitably limited, the codeword produced by the sender is guaranteed to appear in the decoded list, and will appear authentic to the receiver. However, it may not be alone: though the bounded channel cannot produce any *new* authentic codewords, it may be able to cause *prior* ones to appear in the decoded list. This is where the sender's state comes into play: by including the current value of the state in each message, the cryptographic sieve is able to identify the *freshest* authentic word, resulting (with overwhelming probability) in correct, unique decoding.

3.3 Comparison with Related Work

We wish to contrast our results with several other models and techniques for communicating reliably under high error rates.

3.3.1 List Decoding

One of the best-known methods of decoding beyond classical limits is known as *list decoding*, proposed independently by Elias [25] and Wozencraft [70]. In list decoding, a received word is not decoded to a *unique* codeword, but rather to a short *list* of all codewords within a certain radius of the received words, i.e. the encodings of all possible intended messages. (See Section 2.2 for precise definitions and results about list decoding.)

By a probabilistic argument, Guruswami *et al* [36] showed that, for any error rate $\epsilon < 1/2$, there are families of binary error-correcting codes having the following desirable properties:

- their information rate (asymptotically) approaches the Shannon capacity $1 - H(\epsilon)$ of the channel; and
- the list size for error rate ϵ is bounded by a constant (which depends on ϵ, but not on the length of the codewords).

Unfortunately, no efficient list decoding algorithms are yet known for these codes, and in fact their construction is existential (not explicit). On the other hand, for many other well-known codes, there are efficient list-decoding algorithms that can find all codewords within large error radii.

The obvious drawback of list decoding is that one typically wants to recover the *unique* message that was sent, rather than a list of possible messages. The works presented below, as well as our own cryptographic sieve, use list decoding as a tool to extend the decoding radius, but also employ additional techniques in order to identify the correct, unique message in the list.

3.3.2 List Decoding with Side Information

Guruswami [35] designed a coding scheme for binary alphabets that decodes uniquely under adversarial channels with high error rates. The scheme relies on two strong

assumptions (see the discussion below for an analysis of these assumptions):

1. The communicating parties must share a *noise-free* (or low-noise) *side channel* that is used every time a message is sent. (This side channel does not trivialize the solution, because it is only used to send strings that are much shorter than the messages.)

2. The contents of the side channel must remain *secret* from the adversarial channel when it is introducing noise; this imposes either a privacy or timing constraint on the side-channel communication.

Briefly described, Guruswami's scheme works as follows: the sender encodes a message m under a list-decodable code and delivers the encoding x over the noisy channel; simultaneously, the sender selects a random hash function h from a suitable family and transmits the value $y = h(x)$, along with a description of h, over the noiseless side channel. The receiver list-decodes the received word and outputs the codeword z from the decoded list such that $y = h(z)$. For a suitable choice of the hash family and code parameters, the word z from the list will be unique with high probability, and will equal x.

This scheme is very general: it treats the list decoding procedure as a black-box, and decodes uniquely under the same error rate as the list decoding algorithm. The result is unconditional, i.e. no computational assumptions are necessary.

However, there are two disadvantages to the side-channel approach: first, it requires a *noise-free* (or low-noise) side channel, which may not be available in most applications. The second assumption is not stated explicitly, but is actually quite essential: either the side channel must be *private* (unreadable by the adversarial channel), or the side information must be withheld until the adversarial channel delivers its noisy word. (Presumably, this second option adds more communication rounds: the receiver must acknowledge receipt of the message before the side-channel information is sent.) In fact, an adversarial channel can provably defeat the scheme if it learns the side-channel information, by choosing its error pattern to be dependent on the choice of the hash function h and hash value $h(x)$.

3.3.3 Code Scrambling

The *code-scrambling* method of Ding *et al* [21] assumes that the sender and receiver share some random (or pseudorandom) data that is unknown to the channel. This secret randomness is used to "scramble" codewords so that adversarial (Hamming) noise in the scrambled words is equivalent to randomly-distributed (Shannon) noise in the unscrambled words. It therefore suffices to design codes that perform well under high error rates, but for *random* noise.

Both code scrambling and our cryptographic sieve assume the existence of one-way functions.[1] But our underlying model compares favorably to that of code scrambling in a few important ways:

[1] However, the two techniques employ different cryptographic primitives, both of which are implied by one-way functions.

Cryptographic setup. The code-scrambling method requires a random secret key to be shared between the communicating parties and kept secret from the channel. Such a setup requires either the parties to meet in advance (which may be unrealistic), or some *interactive* protocol to establish the private key (in addition, such a protocol would have to deal with the noisy channel that separates the two parties!).

In contrast, our cryptographic sieve works in a *public key* setting: we only require the sender to have a single public key that is known to the recipient. In fact, our scheme works even when the channel possesses all the information available to the receiver, and is potentially even more computationally powerful than the receiver.

Local state and multiple receivers. In reality, two communicating parties usually send and receive many messages over time. Using a classical error-correcting code, this presents no problem: each message is simply encoded and decoded independently. However, when computational assumptions and shared state are introduced, one must be more careful.

The first observation is that in the code-scrambling method, the shared key (or portion thereof) must only be used *one time*. If any part is re-used, the adversary gains information about how the codewords are scrambled, and may be able to introduce "non-random" errors in the future. Therefore, the code-scrambling method requires both parties to keep *synchronized state*. That is, they must always agree on what fresh portion of their shared (pseudo)random key will be used for scrambling and unscrambling the next message. If the parties fall out of sync (say, due to an unexpected transmission or hardware failure), then future messages will decode incorrectly.[2]

Code scrambling also requires a sender to maintain independent secret state for each of the receivers which which it may want to communicate.

In contrast, our cryptographic sieve only requires the sender to maintain a small amount of local state, independent of the receiver or receivers (who are stateless). A sender may keep a single global state variable for use with all receivers. If a message is dropped or ignored, there is no effect on the correctness of future messages. [3]

The feasible-adversary assumption. Code-scrambling works against a feasible channel assuming the existence of a pseudorandom generator (PRG), but this assumption itself is not essential to the solution. The technique works equally well against an unbounded channel, as long as the sender and receiver share a sufficiently long random key. The PRG is used only to decrease the amount of shared randomness that is needed. The computational assumption is therefore an *additional but*

[2]To relax the synchronization requirements, one might imagine sending some "synchronizing information" along with each message. However, the synchronizing information is also subject to errors, so it must be protected by some encoding, and also be recoverable separately from the message. (Using a pseudorandom function for synchronization suffers from a similar problem.) Eliminating the synchrony requirement seems to be quite difficult.

[3]Of course, we cannot provide such a guarantee if the channel is allowed to arbitrarily *delay* messages and *swap* their order — however, neither can code scrambling.

unnecessary one, and does not supplant the assumption of shared secret randomness.

In contrast, our scheme relies on the computational assumption in an *essential* way, due to the permissiveness of the model. In Proposition 3.6.6, we prove that for high error rates, an unbounded channel can *always* defeat any coding scheme in which the receiver is stateless and keeps no secrets from the channel. Because our receiver has exactly these desirable properties, the bounded-channel assumption is necessary. Fortunately, Propositions 3.6.4 and 3.6.5 demonstrate that our scheme also *exploits* the feasible channel to the fullest extent, by working in the weakest possible model.

Universality. Because it relies on the randomness of the errors, the analysis of the code-scrambling method depends essentially upon the properties of the underlying code.[4] The authors also point out that this analysis can be applied to certain algebraic-geometry codes, and that experimental simulations using expander codes have shown positive results. However, each application of code scrambling to a new family of codes requires a new analysis (and yields, potentially, a new set of workable parameters).

Our construction, instead, is fully general: it uses an efficient list-decoding algorithm as a black-box, and requires no other special properties of the code. It reduces *unique decoding* against a *bounded* adversary to *list decoding* against an *unbounded* adversary, while retaining all the asymptotic parameters of the original code.

We also compare the quantitative parameters of our construction with those of code-scrambling:

Binary alphabets. For binary alphabets, code-scrambling yields schemes that tolerate the optimal error rate of $\epsilon = 1/2 - \gamma$ for any $\gamma > 0$. In addition, the information rate is optimal, because it asymptotically approaches the Shannon limit of $1 - H(\epsilon)$.

Our method also provides unique decoding from a $1/2 - \gamma$ error rate. We stress that while our technique yields positive asymptotic information rate, it does not yet match the Shannon limit, because the information rate is dictated by the underlying list-decodable code. While list-decodable codes matching the Shannon limit for any error rate are known to exist [36], it is not known how to efficiently list decode them. Fortunately, progress in list decoding continues rapidly, and any future improvements will automatically apply to our construction.

Large alphabets. When implemented with Reed-Solomon codes (which require large alphabets), code scrambling allows unique decoding from a $\min(1 - \sqrt{R}, 1 - 2R)$ error rate (where R is the information rate), while classical unique decoding of RS codes only allows a $(1 - R)/2$ error rate. Therefore, for $R \geq 1/3$ the code-scrambling

[4] For example, in the analysis of decoding Reed-Solomon codes under random errors, the *maximum distance separability* (MDS) property of the code is the key ingredient. The MDS property does not hold for codes in general; in fact it is quite rare.

method offers no advantage over classical decoding; for $R \in (1/4, 1/3)$ there are some benefits but they are not as pronounced as in the low-rate case.

In contrast, with Reed-Solomon codes our method meets or exceeds the asymptotic error correction rate of the code-scrambling method, at all information rates. In particular, it allows unique decoding from a $1 - \sqrt{R}$ error rate, for all values of $R > 0$. Even better, when implemented with the folded Reed-Solomon construction of Guruswami and Rudra [38], our method tolerates error rates up to $1 - R$, which is optimal for a given information rate.

3.3.4 Private Codes

In an independent and concurrent work, Langberg [46] describes "private codes" in which the sender and recipient use a shared secret random key (which is not known to the channel) to uniquely decode under high error rates.

Langberg assumes a computationally unbounded channel and focuses mainly on existential (rather than efficiently constructible) codes, and on tight bounds for the length of the secret key. The construction uses certain combinatorial set systems to define a "secret subcode" C_r of a given error-correcting code C, which depends on the secret key r. Unique decoding is performed by maximum-likelihood decoding *within* C_r. The analysis and security proof of the scheme are somewhat complex and difficult to penetrate.

Compared to our authentic codes, private codes have a few drawbacks. Namely,

1. *They require secret randomness to be shared between the sender and receiver and kept secret from the channel.* By contrast, in our model the channel is on "equal footing" with the receiver: it knows everything that is known to the latter.

2. *They require the sender and receiver to keep synchronized state,* otherwise they cannot be used to send multiple messages over time. No such requirement exists in our model, and our scheme explicitly handles the goal of multiple messages.

Finally, in our view, we contribute a conceptually cleaner framework, a more general construction, and a modular security proof. In retrospect, it is possible to cast private codes as one *specific instantiation* of our technique combining *message authentication* and *cryptographic sieving.* One can interpret private codes as using a (perfectly secure) symmetric key *message authentication code* (MAC), rather than a (computationally secure) digital signature scheme or MAC. In this interpretation, the "secret subcode" C_r is made up of encodings of pairs (m, σ), where m is an arbitrary message and σ is a valid authentication MAC tag under secret key r. Maximum-likelihood decoding within C_r is accomplished by first list decoding within C, then sieving out the decoded messages and tags that are authentic relative to r.[5]

[5]We thank Adam Smith for pointing out this relationship.

3.3.5 Multicast Authentication

Lysyanskaya, Tamassia, and Triandopoulos [49] used techniques very similar to ours in constructing a *multicast authentication* scheme for *fully-adversarial networks*. They considered the problem of authenticating a set of packets (with as little overhead as possible) in a network that can drop chosen packets, rearrange packets arbitrarily, and insert new packets into the transmitted stream (up to some thresholds). Their solution combines list-decoding of Reed-Solomon codes with cryptographic primitives (specifically, collision-resistant hashing and digital signatures). Implicit in their scheme is a method for uniquely decoding Reed-Solomon codes by encoding authentication tags together with the data payload, and sieving a list of codewords by checking authenticity (as is done in our construction of a coding scheme).

There are some interesting technical differences between the work of Lysyanskaya *et al* and our work: in their multicast scheme, the *packets themselves* are not encoded under a Reed-Solomon code — only the *hashes* of the packets are. This suffices in their setting because the requirements are weaker: the receiver is not obligated to recover all the data that was sent (because the network is allowed to drop packets), but only the packets that were actually delivered to it. The use of list-decoding and sieving is limited to the authentication information, which is used to check the authenticity of packets that are actually delivered. Also, in the multicast scheme, the sender and receivers share a small amount of *public state*, called the "group identification tag," which can never be re-used. In contrast, our scheme only requires the sender to maintain a counter or clock, which has semantics that allow a stateless receiver to disambiguate among messages without being synchronized with the sender. It appears that this technique can be applied to the multicast scheme in order to relax its synchrony requirements.

3.4 Formal Model and the Coding Game

The theory of error-correcting codes is concerned mainly with the combinatorial and algebraic properties of codes (such as distance, alphabet size, rate, and linearity). In many cases, the theory is also concerned with the computational complexity of encoding and decoding procedures. However, in the past there has been no need to address issues such as the complexity of channels, the source of encoded messages, stateful players, randomization, and cryptographic keys. All of these issues are relevant to our work, and they require careful and precise modelling for several reasons: first, so that the assumptions underlying our scheme can be fairly evaluated. Second, so that we can rigorously analyze and prove correctness of our coding schemes. Third, so that we know precisely what our schemes can and cannot deliver in different applications and settings.

In this section, we motivate and precisely define a formal model for a feasible adversarial channel, and give the properties that a correct coding scheme should satisfy. As is standard in the cryptographic literature, we do this by describing a *game* or *experiment* in which an adversary interacts with a scheme in some prescribed manner,

then we specify the desired outcome of the experiment. We call this experiment the *coding game*, and the desired outcome of the game will define a *robust coding scheme.*

Motivation and discussion. In formulating our model and requirements for a robust coding scheme, there are several considerations we must take into account. Below we discuss some of these considerations, and the motivations behind our decisions.

We prefer a coding scheme that has the weakest possible cryptographic setup requirements. In practice, a public-key setup is easier and more flexible, and arguably less risky than one in which parties share a common secret key.[6] Also, it is natural that the entity that will be sending data should also be the one whose key is published openly. Therefore in our model will have the sender generate a public and secret key, with the secret key kept to himself (or at least kept secret from the channel), and the public key known (without errors) to the receiver and the channel.

In many applications of a coding scheme, we cannot predict the source of messages to be encoded. In a real application, it may be that the adversary has partial or even total influence over the messages. Furthermore, it may be that it is easier for a channel to cause confusion by corrupting certain messages, or by seeing the encodings of certain messages.[7] Therefore in order to make the coding scheme as widely applicable as possible, we make the safest possible assumption: that the channel has complete, adaptive control over the choice of messages that are encoded.

We would also like our scheme not to impose any synchrony requirements on the sender and receiver, which are independent entities that may be separated by a large distance. In real applications, a receiver may want to (or need to) ignore certain messages while "offline," for example due to maintenance, loss of power, or disconnection from the communication medium. The receiver should not be required, nor may it be possible, to keep track of what messages (or even how many messages) have been transmitted while offline. We also cannot anticipate what may cause a receiver to go offline. Hence, we will again make the most conservative choice, and leave it to the adversarial channel to decide when the receiver will be "online," i.e. which (noisy) encoded messages it will receive. An equivalent way of describing this choice is to let the channel arbitrarily decide to *drop* messages whenever it wants to, leaving the receiver unaware that those messages were encoded at all. Of course, it is too much to require the receiver to correctly decode a message that it has not even received, therefore our definition will not impose any requirements on the receiver when a message is dropped. However, the receiver *will* be required to gracefully recover from dropped messages, by correctly decoding all delivered messages thenceforth (even though it is unaware that messages have been dropped in the meantime).

Definition 3.4.1 (Coding scheme and channel)**.** A *coding scheme* $\mathcal{CS}$ with *block lengths* $n(k)$ and *alphabets* Σ_k (for $k \in T$, an infinite subset of $\mathbb{N}$) is a triple of

[6] However, two may keep a secret, if one of them is dead (with apologies to Benjamin Franklin).

[7] Many similar considerations apply in the motivation for chosen-message security in authentication and chosen-ciphertext security in encryption. Indeed, our definition most closely resembles the definition of existential unforgeability under a chosen-message attack for digital signatures or message authentication codes.

probabilistic, polynomial-time interactive Turing machines $\mathcal{CS} = (G, S, R)$ having the properties below.

For ease of notation, we write $n = n(k)$ and $\Sigma = \Sigma_k$ when the choice of $k \in T$ is clear by context. Then:

- On input 1^k, G outputs a pair (pk, sk), where by convention $pk = (1^k, pk')$ and $sk = (pk, sk')$.[8]
- Given (pk, sk) and on input $m \in \Sigma^k$, S outputs some $x \in \Sigma^n$.
- Given pk and on input $r \in \Sigma^n$, R outputs an element of Σ^k.

The *information rate* of such a scheme is defined to be $\liminf_{k\to\infty} \frac{k}{n(k)}$.

An (adversarial) *channel* $\mathcal{C}$ introducing $e(k)$ *(absolute) errors* for a coding scheme $\mathcal{CS}$ is a family of randomized, interactive circuits $\{\mathcal{C}_k\}_{k\in T}$.

The coding game. A channel interacts with a coding scheme in the *coding game*, which proceeds as formally described in Figure 3-1, and as depicted pictorially in Figure 3-2. The game works roughly as follows: in an initial set-up phase, the key generation algorithm G produces public and secret keys, which are delivered (without errors) to the appropriate algorithms. The channel $\mathcal{C}$ then adaptively queries the sender S to encode messages of the channel's choice. For each encoded message, the channel (at its option) either introduces noise and delivers the noisy version to the receiver R, or "drops" the message entirely (without the receiver even knowing that a message was encoded). If the channel elects to deliver a noisy block, the receiver is obligated to decode the message, and outputs its best guess (which the channel can see). The channel succeeds if it is able to cause the receiver to output a message different from the intended one in any of the iterations in which the channel did not drop the message.

Channel advantage and robust coding schemes. The following definitions capture the notion that a robust coding scheme should always behave correctly in the presence of a feasible channel, except with some small probability of failure.

Definition 3.4.2 (Advantage in the coding game). Let $\mathcal{C} = \{\mathcal{C}_k\}$ be a channel for coding scheme $\mathcal{CS}$ in the coding game. Define

$$\mathsf{Adv\text{-}CG}_{\mathcal{C}}^{\mathcal{CS}}(k) = \Pr[\mathcal{C}_k \text{ succeeds against } \mathcal{CS} \text{ for message length } k],$$

where the probability is taken over the random coins of $\mathcal{CS}$ and $\mathcal{C}$.

For functions $q(k), s(k), e(k)$, define

$$\mathsf{Adv\text{-}CG}_{q,s,e}^{\mathcal{CS}}(k) = \max_{\mathcal{C}} \left\{\mathsf{Adv\text{-}CG}_{\mathcal{C}}^{\mathcal{CS}}(k)\right\},$$

[8]The string 1^k is interpreted to mean the symbol 1, repeated k times (not 1 raised to the kth power).

Let $\mathcal{CS} = (G, S, R)$ be a coding scheme with block lengths $n(k)$ and alphabets Σ_k for $k \in T$, and let $\mathcal{C} = \{\mathcal{C}_k\}_{k \in T}$ be a channel introducing $e(k)$ absolute errors. For convenience of notation, write $n = n(k)$, $\Sigma = \Sigma_k$, and $e = e(k)$ when k is clear from context.

The *coding game* (for message length $k \in T$) between the channel and the coding scheme proceeds as follows:

1. On input 1^k, G produces (pk, sk). S receives sk as input, while $\mathcal{C}_k$ and R receive pk as input.
2. The following process is iterated until $\mathcal{C}_k$ terminates:
   (a) On the jth iteration, $\mathcal{C}_k$ chooses a message $m_j \in \Sigma^k$ and gives it to S.
   (b) S outputs $x_j \in \Sigma^n$ and gives it to $\mathcal{C}_k$.
   (c) $\mathcal{C}_k$ chooses one of the following two options:
      i. $\mathcal{C}_k$ computes $r_j \in \Sigma^n$ such that $\Delta(x_j, r_j) \leq e$, and gives it to R.
      ii. $\mathcal{C}_k$ provides no input to R.
   (d) In case 2(c)i, R outputs some $m'_j \in \Sigma^k$.
   In case 2(c)ii, we define $m'_j = m_j$.

The channel *succeeds* in the game against $\mathcal{CS}$ if $m'_j \neq m_j$ for any iteration j.

Figure 3-1: Formal definition of a channel interacting with a coding scheme in the coding game, and the notion of a successful channel.

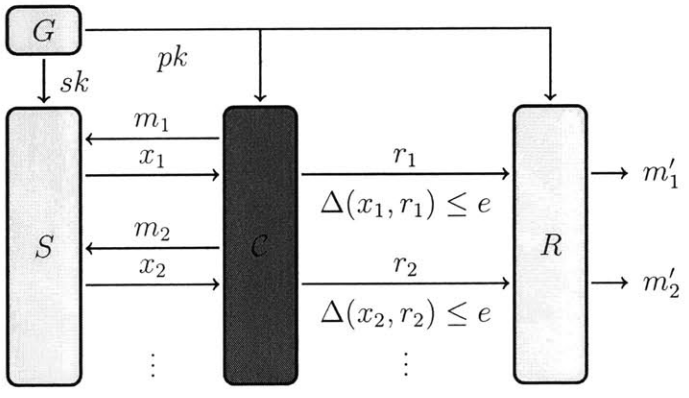

Figure 3-2: Pictorial representation of the coding game. Time flows downward.

where the maximum is taken over all $\mathcal{C} = \{\mathcal{C}_k\}$ such that $\mathcal{C}_k$:

1. queries the sender to encode at most $q(k)$ messages in any execution of the coding game (for message length k), and

2. has circuit size at most $s(k)$, and

3. introduces at most $e(k)$ absolute errors.

Definition 3.4.3 (Robust coding scheme)**.** A coding scheme $\mathcal{CS}$ is called *robust under $e(k)$ errors* if, for every $q(k), s(k) \in \text{poly}(k)$, $\mathsf{Adv\text{-}CG}^{\mathcal{CS}}_{q,s,e}(k)$ is negligible in k.

Technical remarks on the definitions. Definition 3.4.1 of a coding scheme requires the scheme to be explicitly polynomial-time "constructible," in a sense analogous to that for a family of error-correcting codes. In particular, the sender S is a uniform poly-time algorithm that can encode messages of any length $k \in T$, and the receiver R is a uniform poly-time algorithm that can decode messages of any block length $n(k)$.

On the other hand, in Definition 3.4.1 we allow the channel to be a non-uniform algorithm, by modelling it as a family of circuits. In the definition of a channel itself, we do not impose any constraints on the circuit complexity or query complexity of the channel. Instead, we parameterize the success of a channel by these complexities (as explicit functions in k) in Definition 3.4.2.

In the definition of the coding game, the channel does *not* explicitly see the output of the receiver R in Step 2d. This does not weaken the adversary, for the following reason: either R produces an incorrect output $m'_j \neq m_j$, in which case the channel is immediately successful (and will remain so until the end of the game), or R produces the correct output $m'_j = m_j$, which the channel already knows (because it generated m_j itself). We find it easier to analyze our security reductions without having to account for this extra input to the channel.

As we have defined a coding scheme and channel advantage, the parameter k serves a dual purpose: it acts as both the message length and as the cryptographic notion of "security parameter" for the coding scheme (i.e., the parameter relative to which an adversary should have negligible advantage). This is a natural choice, because our constructions must take asymptotics into account on both the coding theory side and the cryptography side, and these asymptotics are dependent on each other. Our constructions will require families of error correcting codes that are *poly-time constructible*; this notion is defined relative to the message length. At the same time, our cryptographic reductions require an explicit link between the security parameter of the underlying cryptographic primitive and the message length of the coding scheme.

For formal definitions of interaction among algorithms/circuits, we refer the reader to the standard notion of interactive Turing machines [33, 15, 14].

3.5 Constructing Robust Coding Schemes

3.5.1 Intuition

One first attempt at constructing a coding scheme is to encode each message, sign the encoding, and send the signature along with it. At the receiver, the noisy encoding can be list-decoded, and the signature verification can serve as a cryptographic sieve to select the correct message from the list. Unfortunately, this simple scheme does not work, because the signature can be destroyed by errors. The natural fix is to protect the signature as well by encoding it under a suitable code. Unfortunately, a careful analysis (even using list decodable codes for both the message and signature) shows that this approach yields no improvement in overall rate.

The key insight is that the message and signature should be encoded *together*, rather than separately. To communicate a message m, the sender can first sign m, yielding a signature σ, then encode the pair (m, σ) as a unit. To decode a received word r, the recipient can list decode r, yielding a list of potential pairs (m_i, σ_i). If the number of errors is suitably bounded, then the original pair (m, σ) will appear in the list, and will be declared authentic by the signature verifier. By the unforgeability of the signature scheme, the other pairs should be declared inauthentic and can be thrown out, leaving only the original message as the unique output.

There is one hidden difficulty in the above description: the list may actually include *several* authentic (m_i, σ_i) pairs, which correspond to messages sent *in the past*. This event cannot be used to break the signature scheme, because a valid forgery must involve a *new*, previously-unsigned message. Fortunately, it suffices to find a way to correctly disambiguate among several valid pairs in the list.

The solution is for the sender to maintain some local state in the form of a short *counter* or *clock* t. With each message, the current value of t is appended to the message (and signed along with it), then incremented (or is incremented automatically over time, as in the case of a clock). Then at the receiving end, among all valid pairs in the decoded list, the recipient chooses the one with the largest value of t. In other words, the receiver always decodes to the "freshest" authentic message in the list. We note that the this receiver is *stateless*, while the sender needs only to maintain the current value of the counter t.

There are only two essential requirements for the counter values: each value must be *used at most once* (a "nonce"), and the values must be *ordered* so that the receiver is able to recognize the largest value in a list. Any monotonically increasing sequence of integers satisfies these requirements; in particular, a timestamp is sufficient. We stress that the sender's clock need not be synchronized with any other clock in the world, nor is clock "drift" a problem.

Looking at the entire construction from an informal (and perhaps amusing) level, it may be viewed as an affirmation of a basic rule of etiquette: one should always *date* and *sign* one's correspondence!

3.5.2 Formal Construction

In this section we formally construct a coding scheme according to the intuition given above.

Required primitives. Let $\mathbb{C} = \left\{C_\kappa : \Gamma_\kappa^\kappa \to \Gamma_\kappa^{\eta(\kappa)}\right\}_{\kappa \in \mathbb{N}}$ denote a family of error-correcting codes[9] having the following properties (see Section 2.2 for the relevant coding theory background):

- $\mathbb{C}$ is an explicit poly-time constructible family with encoding algorithm $\mathcal{E}$;
- $\mathbb{C}$ is $(\xi(\kappa), L(\kappa))$-list decodable via a polynomial-time algorithm $\mathcal{LD}$.

Therefore $\mathbb{C}$ has information rate R, where

$$R = \liminf_{\kappa \to \infty} \frac{\kappa}{\eta(\kappa)}.$$

Let $\mathcal{SS} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a digital signature scheme (see Section 2.3 for definitions). As a convention of notation, we will invoke $\mathcal{S}$ and $\mathcal{V}$ on inputs encoded over an appropriate alphabet Σ_k; likewise, the output of $\mathcal{S}_{sk_S}$ will be interpreted as a string over Σ_k. Without loss of generality, we will invoke the key generator $\mathcal{G}$ with a security parameter of the form $k' = \ell \cdot \lfloor \lg |\Sigma_k| \rfloor$, so that the signatures produced by $\mathcal{S}$ can be interpreted as elements of Σ_k^ℓ.

The coding scheme. We now define the coding scheme $\mathcal{CS} = (G, S, R)$. The scheme is actually parameterized by an integer function $\ell(k)$, which relates the lengths of the signature and counter (written over appropriate alphabet Σ_k) that are appended to a message of length k in our scheme; thus $\ell(k)$ determines the *loss* of information rate of our coding scheme relative to the underlying code.

For messages of length k, our scheme will employ the code C_κ for $\kappa = k + 2\ell(k)$. That is, our scheme will use alphabet $\Sigma_k = \Gamma_\kappa = \Gamma_{k+2\ell(k)}$ and block length $n(k) = \eta(\kappa) = \eta(k+2\ell(k))$. As we will show, it will be robust under $e(k) = \xi(\kappa) = \xi(k+2\ell(k))$ errors. The scheme is formally specified below, and also represented pictorially (with some loss in precision) in Figure 3-3. We write $\ell = \ell(k)$, $\Sigma = \Sigma_k$, $n = n(k)$, and $\kappa = k + 2\ell$ where k is clear from context.

G: On input 1^k for $k \in \mathbb{N}$, let $k' = \ell \cdot \lfloor \lg |\Sigma| \rfloor$, and let $(pk_S, sk_S) \leftarrow \mathcal{G}(1^{k'})$.

Output public key $pk = (1^k, 1^\ell, pk_S)$ and secret key $sk = (pk, sk_S)$.

S: Let $t \in \Sigma^\ell$ be a local variable that is initialized to zero and maintained across invocations of S.

Given secret key $sk = (pk, sk_S)$ and on an input message $m \in \Sigma^k$:

[9] We use dummy variable κ, alphabet symbol Γ, and block length function $\eta(\kappa)$ in order to reserve k, Σ, and $n(k)$ for our coding scheme.

1. Let $\sigma \leftarrow \mathcal{S}_{sk_S}(m,t)$, where $\sigma \in \Sigma^\ell$.
2. Let $w = (m,t,\sigma) \in \Sigma^\kappa$.
3. Increment t in a canonical way, e.g. according to lexicographic ordering on its representation as an element of Σ^ℓ.
4. Output $x = C_\kappa(w) = \mathcal{E}(w) \in \Sigma^n$.

R: Given public key $pk = (1^k, 1^\ell, pk_S)$ and on an input block $r \in \Sigma^n$:

1. List-decode r, interpreting each item in the list as the encoding of a triple $(m^{(i)}, t^{(i)}, \sigma^{(i)}) \in \Sigma^\kappa$. More precisely, let

$$\left\{(m^{(i)}, t^{(i)}, \sigma^{(i)})\right\}_{i \in [L(\kappa)]} \leftarrow \mathcal{LD}(r),$$

   where $m^{(i)} \in \Sigma^k$ and $t^{(i)}, \sigma^{(i)} \in \Sigma^\ell$ for all $i \in [L(\kappa)]$.
2. Find all authentic triples

$$V = \left\{(m^{(i)}, t^{(i)}, \sigma^{(i)}) \ : \ \mathcal{V}_{pk_S}((m^{(i)}, t^{(i)}), \sigma^{(i)}) = 1\right\}.$$

3. Find the largest authentic value of t:

$$t^{\max} = \max\left\{t^{(i)} \ : \ (m^{(i)}, t^{(i)}, \sigma^{(i)}) \in V\right\}.$$

4. Find all authentic messages having the largest authentic value of t:

$$M = \left\{m^{(i)} \ : \ (m^{(i)}, t^{\max}, \sigma^{(i)}) \in V\right\}.$$

5. If $M = \{m\}$, a singleton set, output m. N.B.: there may be many authentic triples of the form $(m^{(i)}, t^{\max}, \sigma^{(i)})$; if all such $m^{(i)}$ are the same (even if their $\sigma^{(i)}$ differ), M is a singleton set.
   Otherwise — i.e., there are two or more *different* authentic messages $m^{(i)}$ having counter value $t^{\max}$ — output $\perp$.

3.5.3 Proof of Robustness

Theorem 3.5.1. *Let $\mathbb{C}$, $\mathcal{SS}$, and $\mathcal{CS}$ be as described above, where $\mathcal{CS}$ is parameterized by a function $\ell(k)$. Let $q(k), s(k)$ be arbitrary functions. Then for any k such that $q(k) < |\Sigma_k|^{\ell(k)}$, we have:*

$$\mathsf{Adv\text{-}CG}^{\mathcal{CS}}_{q,s,e}(k) \leq \mathsf{Adv\text{-}CMA}^{\mathcal{SS}}_{q',s'}(k')$$

where

$$k' = \ell(k) \cdot \lfloor \lg |\Sigma_k| \rfloor, \qquad e(k) = \xi(k + 2\ell(k)), \qquad q'(k') = q(k),$$
$$s'(k') = s(k) + q(k) \cdot (\mathsf{time}_{\mathcal{S}}(k') + \mathsf{time}_{\mathcal{LD}}(k) + L(k + 2\ell(k)) \cdot \mathsf{time}_{\mathcal{V}}(k')) + O(q(k) \cdot (k + 2\ell(k)) \log |\Sigma_k|))$$

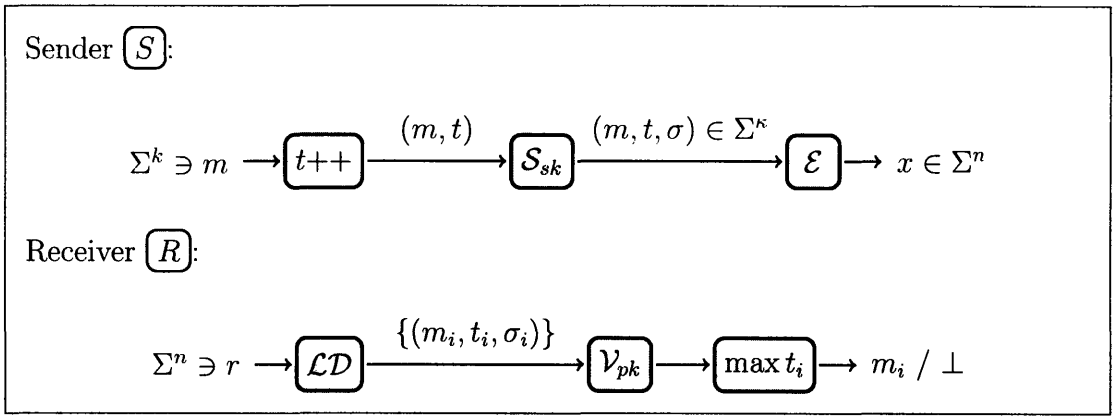

Figure 3-3: Pictorial representation of the sender and receiver in the coding scheme.

Proof. Let $\mathcal{C} = \{\mathcal{C}_k\}_{k\in\mathbb{N}}$ be any channel for coding scheme $\mathcal{CS}$ that queries the sender to encode at most $q(k)$ messages, has circuit size at most $s(k)$, and introduces at most $e(k)$ absolute errors. Let k be such that $q(k) < |\Sigma_k|^{\ell(k)}$.

We will use $\mathcal{C}$ to build a forger $\mathcal{F} = \{\mathcal{F}_{k'}\}$ whose advantage in generating a forgery (via a chosen-message attack) against $\mathcal{SS}$ is at least as large as $\mathcal{C}$'s advantage in the coding game against $\mathcal{CS}$.

Recall that in a chosen-message attack, keys (pk_S, sk_S) are generated by $\mathcal{G}(1^{k'})$. Then $\mathcal{F}_{k'}$ is given pk_S as input, and has oracle access to $\mathcal{S}_{sk_S}$. Our forger $\mathcal{F}_{k'}$ will run $\mathcal{C}_k$ internally, emulating a view of the coding game to $\mathcal{C}_k$.

$\mathcal{F}_{k'}$ is formally defined in Figure 3-4 (note that the structure mirrors that of the coding game from Figure 3-1).

Complexity of the forger. It is clear that the number of signature queries is at most $q(k)$. Furthermore, a direct analysis shows that the size of $\mathcal{F}_{k'}$ is as claimed in the theorem statement: $\mathcal{F}_{k'}$ incorporates one copy of $\mathcal{C}_k$, $q(k)$ copies of the signature algorithm $\mathcal{S}$ and list-decoding algorithm $\mathcal{LD}$, and $q(k) \cdot L(k + 2\ell(k))$ copies of the signature verification algorithm $\mathcal{V}$, plus incidental "glue" circuitry to connect those components.

The output of the forger. It is easy to see that whenever $\mathcal{F}$ outputs some $((m^{(i)}, t^{(i)}), \sigma^{(i)})$, it is a valid forgery for an unqueried message $(m^{(i)}, t^{(i)})$: indeed, $\mathcal{V}_{pk_S}((m^{(i)}, t^{(i)}), \sigma^{(i)}) = 1$, and $(m^{(i)}, t^{(i)})$ was not a query to $\mathcal{S}_{sk_S}$ because (1) if $t^{(i)} > t$, no query's second component is as large as $t^{(i)}$, and (2) if $t^{(i)} = t$, the unique query whose second component was t had a first component $m_j \neq m^{(i)}$. (Here we have used the fact that $q(k) < |\Sigma_k|^{\ell(k)}$, so t never exhausts its domain of possible values during the execution of $\mathcal{F}$.)

The advantage of the forger. First we observe at any point prior to $\mathcal{F}$ producing a forgery and halting, the view it provides to its internal copy of $\mathcal{C}$ is distributed identically to the view seen by $\mathcal{C}$ in the coding game against $\mathcal{CS}$. Indeed, the public

Assume that the appropriate values of k and $\ell = \ell(k)$ are "hard-coded" into the circuit $\mathcal{F}_{k'}$. Let $\Sigma = \Sigma_k$, $\mathcal{C} = \mathcal{C}_k$, and $n = n(k)$. Let $t \in \Sigma^\ell$ be a local variable that is initialized to zero.

1. On input pk_S, create $pk = (1^k, 1^\ell, pk_S)$ and give pk to $\mathcal{C}$. (The values k and $\ell(k)$ can simply be "hard-coded" into the circuit $\mathcal{F}_{k'}$.)

2. Iterate until $\mathcal{C}$ terminates (as in the coding game):

   (a) On the jth iteration, $\mathcal{C}$ outputs a message $m_j \in \Sigma^k$ for the sender S.

   (b) Emulate the sender S in the following way:
   Let $t_j = t$.
   Query the signing oracle $\mathcal{S}_{sk_S}$, letting $\sigma_j \leftarrow \mathcal{S}_{sk_S}(m_j, t_j)$.
   Let $x_j = C_\kappa(m_j, t_j, \sigma_j) = \mathcal{E}(m_j, t_j, \sigma_j)$.
   Increment t, and give x_j to $\mathcal{C}$.

   (c) $\mathcal{C}$ chooses one of two options:
   i. $\mathcal{C}$ outputs (for R) an $r_j \in \Sigma^n$ such that $\Delta(x_j, r_j) \leq e(k)$.
   ii. $\mathcal{C}$ provides no output for R.

   (d) In case 2(c)i, emulate the receiver R (to a point) in the following way:
   - List-decode r_j: $\left\{(m^{(i)}, t^{(i)}, \sigma^{(i)})\right\} \leftarrow \mathcal{LD}(r_j)$.
   - Find authentic triples
   $$V = \left\{(m^{(i)}, t^{(i)}, \sigma^{(i)}) \ : \ \mathcal{V}_{pk_S}((m^{(i)}, t^{(i)}), \sigma^{(i)}) = 1\right\}.$$
   - If there exists $(m^{(i)}, t^{(i)}, \sigma^{(i)}) \in V$ such that either:
     - $t^{(i)} > t_j$, or
     - $t^{(i)} = t_j$ and $m^{(i)} \neq m_j$,

     output $((m^{(i)}, t^{(i)}), \sigma^{(i)})$ as a forgery and halt.
   - Otherwise, continue.

   In case 2(c)ii, just continue.

3. Halt with no output.

Figure 3-4: Definition of the forger algorithm $\mathcal{F}_{k'}$.

key pk and every encoding x_j are produced in exactly the same way as in the coding game, and these are the only components of the channel's view. In particular, it follows that in any iteration of the coding game, the set V constructed by R is distributed identically to the set V constructed in the corresponding iteration when $\mathcal{F}$ emulates R. Therefore in the following, we will simply refer to V in some iteration, referring to both the coding game and the forger's emulation.

We now argue that any event in which R's output in iteration j of the coding game is not m_j corresponds exactly to an event in which the forger halts with a (valid) forgery. This will establish the claimed inequality on the advantages.

Consider iteration j: first note that in the emulation (and hence in the coding game), $(m_j, t_j, \sigma_j) \in V$, because the channel $\mathcal{C}$ introduces at most $e(k)$ errors, and by the correctness of $\mathcal{CD}$. By implication, R calculates $t^{\max} \geq t_j$. Now suppose that R does not output m_j in iteration j of the coding game. There are two cases:

1. R outputs $m'_j \in \Sigma_k^k$ with $m'_j \neq m_j$: this can happen only when $t^{\max} > t_j$, so there is some $(m, t, \sigma) \in V$ with $t > t_j$. But this causes $\mathcal{F}$ to halt with a forgery in the emulation.

2. R outputs $\perp$: this implies that there are $(m^{(a)}, t^{\max}, \sigma^{(a)}), (m^{(b)}, t^{\max}, \sigma^{(b)}) \in V$ with $m^{(a)} \neq m^{(b)}$.

   Then either $t^{\max} > t_j$, in which case $\mathcal{F}$ halts with a forgery in the emulation, or $t^{\max} = t_j$, in which case either $m^{(a)}$ or $m^{(b)}$ differs from m_j, so $\mathcal{F}$ also halts with a forgery.

This completes the proof. □

Corollary 3.5.2. *Let $\mathbb{C}$, $\mathcal{SS}$, and $\mathcal{CS}$ be as above (where $\mathbb{C}$ has information rate R). Let $\ell(k) = o(k)$ be such that there exists a constant $\delta > 0$ with $\ell(k) \cdot \lfloor \lg |\Sigma_k| \rfloor \geq k^\delta$ for all sufficiently large k.*

If $\mathcal{SS}$ is an existentially unforgeable signature scheme (under chosen-message attack), then $\mathcal{CS}$ is a coding scheme with information rate R that is robust under $e(k) = \xi(k + 2\ell(k))$ absolute errors.

Proof. First we analyze the information rate of the scheme, which is:

$$\liminf_{k\to\infty} \frac{k}{n(k)} = \liminf_{k\to\infty} \frac{k}{\eta(k + 2\ell(k))} = \liminf_{k\to\infty} \frac{Rk}{k + 2\ell(k)} = \liminf_{k\to\infty} \frac{Rk}{k + o(k)} = R.$$

Now for any polynomially-bounded $q(k), s(k)$, we need to show that

$$\mathsf{Adv\text{-}CG}^{\mathcal{CS}}_{q,s,e}(k)$$

is negligible in k. First, note that for all sufficiently large k,

$$q(k) < 2^{k^\delta} \leq 2^{\ell(k) \cdot \lg |\Sigma_k|} = |\Sigma_k|^{\ell(k)}.$$

Then Theorem 3.5.1 applies, implying that $\mathsf{Adv\text{-}CG}$ is at most $\mathsf{Adv\text{-}CMA}^{\mathcal{SS}}_{q',s'}(k')$, where $k', q'(k'), s'(k')$ are as in the Theorem. Clearly $q'(k')$ and $s'(k')$ are bounded

by polynomials in k', because $\mathcal{S}$, $\mathcal{V}$, and $\mathcal{LD}$ are poly-time algorithms. Then because $\mathcal{SS}$ is existentially unforgeable, $\mathsf{Adv\text{-}CMA}^{\mathcal{SS}}_{q',s'}(k')$ is negligible in k'. Because $k' = \ell(k) \cdot \lfloor \lg |\Sigma_k| \rfloor = \Omega(k^\delta)$, that advantage is also negligible in k, as desired. □

3.5.4 Concrete Instantiations

Here we instantiate our construction with concrete choices of list-decodable codes, yielding robust coding schemes for a variety of good parameter sets.

One issue in common with all the instantiations is how to choose the signature length $\ell(k)$ as a function of the message length k. There is a tradeoff between the information rate of the resulting scheme and its exact security. If security is of the utmost importance, then one can choose $\ell(k) = \delta \cdot k$ for some small $\delta > 0$. This yields a *linearly-preserving* security reduction (in the terminology of Goldreich [28]), but an information rate $R \cdot \frac{1}{1+2\delta}$ (where R is the rate of the underlying code). Alternately, one can choose $\ell(k) = k^\delta$ for some $\delta \in (0, 1)$, which yields a *polynomially-preserving* reduction (in the terminology of Goldreich [28] and Levin, as presented by Luby [48]), but with an asymptotic information rate of R.

Binary coding schemes. To construct coding schemes over the binary alphabets $\Sigma_k = \{0, 1\}$ (for all k), we can use the poly-time constructible and efficiently list-decodable (concatenated) codes of Guruswami and Sudan [40]. By concatenating Reed-Solomon codes with inner codes of sufficiently large distance, they construct codes that are efficiently list decodable under $e = (\frac{1}{2} - \gamma) \cdot n$ errors (for any constant $\gamma > 0$) that have block length $\eta = O(\kappa/\gamma^8)$, i.e. the information rate $R = \Omega(\gamma^8)$. A more complicated construction that concatenates algebraic-geometry codes with the Hadamard code yields information rate $R = \Omega(\gamma^6 \log \frac{1}{\gamma})$. Further work by Guruswami *et al* [36] gave codes of rate $\Omega(\gamma^4)$ having list size $O(\gamma^{-2})$.

Coding schemes for larger alphabets. With the use of larger alphabets (whose sizes grow with κ), we can list-decode under much larger numbers of errors.

The popular family of Reed-Solomon codes [57] was shown by Sudan [66], and later Guruswami and Sudan [39], to be efficiently list decodable under high error rates. Specifically, there is a polynomial-time algorithm that can find all codewords within distance $\eta - \sqrt{\eta\kappa} = (1 - \sqrt{R})\eta$ of any word. The number of words in the list is at most η^2, and the alphabet size is $|\Sigma_\kappa| \approx \eta$.

The recent construction by Guruswami and Rudra [38] of *Folded Reed-Solomon* codes admits list-decoding algorithms with even better performance, correcting from $(1 - R - \gamma)\eta$ errors for any $\gamma > 0$. These codes are called "capacity-achieving," because the relationship between their list-decoding radius and information rate is (nearly) optimal.

Other considerations. There are a variety of other efficiently list-decodable codes in the literature having other interesting properties. For example, the codes of Guruswami and Indyk [37] have encoding and list-decoding algorithms that run in time

linear in the block length, and recover from up to $(1-1/q-\gamma)\eta$ errors with a positive information rate and list size depending only on γ (here q is the alphabet size).

3.5.5 Optimism and Optimizations

In the scheme described in Section 3.5.2, the receiver's algorithm is not as efficient as it could be. The main practical bottlenecks are that the receiver must do (potentially expensive) list decoding, then must perform as many signature verifications as there are codewords in the decoded list. In this section we show how the computational burden on the receiver can sometimes be reduced, via (1) *optimism* about the channel's behavior and (2) certain *optimizations* to the receiver algorithm.

Optimistic decoding. In cryptography, *optimism* is a principle that allows protocols to be *more efficient when the adversary does not use its full powers*, without compromising security in cases in which the adversary does use all its powers. Such protocols are called "optimistic" because their normal mode of operation tends to be simple (and hence efficient), with the "hope" that the adversary will not interfere maliciously. If the adversary does interfere, however, the protocol is designed to detect such interference, and reverts to a more "pessimistic" (and inefficient) mode to complete its task securely.

In our coding scheme, the receiver can employ very efficient optimistic decoding, which avoids the less-efficient list-decoding and signature verification algorithms entirely. The central idea is to first run an *unambiguous decoding* algorithm for the underlying error-correcting code, to test whether the channel has added much noise to the transmitted codeword. If the received word is very close to the decoded codeword (and far from all other codewords), then one can be sure that the channel introduced very little noise, and that the decoded codeword is the intended one.

Our construction $\mathcal{CS}$ from Section 3.5.2 uses an underlying error-correcting code $\mathbb{C}$. In addition to its parameters and list-decoding properties as described above, we will require some further properties:

- C_κ has (Hamming) distance at least $d(\kappa)$.
- $\mathbb{C}$ has a polynomial-time *unambiguous decoding* algorithm $\mathcal{D}$, which, given $r \in \Sigma_\kappa^{\eta(\kappa)}$, computes $c \in C_\kappa$ such that $\Delta(r,c) \leq d(\kappa)/2$ if such a c exists.

Our optimistic decoding algorithm R' now works as follows (we use the same notation as in our formal description of $\mathcal{CS}$ above):

R': Given public key $pk = (1^k, 1^\ell, pk_S)$ and on input block $r \in \Sigma^n$:

1. Attempt to unambiguously decode r: let $c \leftarrow \mathcal{D}(r)$. If $\mathcal{D}$ produces no output, run the "pessimistic" algorithm R from above.
2. If $\Delta(r,c) < d(\kappa) - e(\kappa)$, interpret c as the encoding of a triple (m,t,σ), and output m. Otherwise, run the pessimistic algorithm R from above.

Note that R' completely elides verification of the signature in the triple (m,t,σ).

Proposition 3.5.3. *Let $\mathcal{CS}' = (G, S, R')$ be an optimistic version of the coding scheme $\mathcal{CS}$ from above. Then Theorem 3.5.1 applies with $\mathcal{CS}'$ replacing $\mathcal{CS}$.*

Proof. It suffices to show that when R' outputs some m without reverting to the pessimistic R, the intended message was m.

Suppose that in an iteration of the coding game, S outputs a codeword $x \in C_\kappa$, the channel delivers a word r, and R' computes a $c \in C_\kappa$ such that $\Delta(r, c) < d(\kappa) - e(\kappa)$. By assumption on the channel, $\Delta(x, r) < e(\kappa)$, so by the triangle inequality we have:

$$\Delta(x, c) \leq \Delta(x, r) + \Delta(r, c) < d(\kappa).$$

Because both $x, c \in C_\kappa$, we must have $c = x$. Therefore R' outputs the intended message. □

To get an idea of the error rates for which optimistic decoding can be useful, we briefly analyze the coding scheme when instantiated with Reed-Solomon codes. These codes have distance $d(\kappa) = \eta(\kappa) - \kappa + 1 \approx n(1 - R(\kappa))$, where κ is the message length, $\eta(\kappa)$ is the block length, and $R(\kappa) = \frac{\kappa}{\eta(\kappa)}$ is the information rate. The Berlekamp-Massey algorithm [6] can be used for unambiguous decoding to distance $d(\kappa)/2$, and the algorithm of Guruswami and Sudan [39] can be used to list-decode to distance $e(\kappa) = \eta(\kappa) - \sqrt{\eta(\kappa) \cdot \kappa} = n(1 - \sqrt{R(\kappa)})$. Therefore our coding scheme can recover from $e(\kappa)$ errors, and the optimistic decoding algorithm works when the channel introduces at most

$$d(\kappa) - e(\kappa) - 1 = \sqrt{\eta(\kappa) \cdot \kappa} - \kappa = n(\sqrt{R(\kappa)} - R(\kappa))$$

errors. For a rate $R(\kappa) = \frac{1}{100}$, optimistic decoding applies when the channel introduces at most $\frac{9}{100}n$ errors, whereas pessimistic decoding always recovers from up to $\frac{9}{10}n$ errors. More generally, optimistic decoding works whenever the channel introduces at most a factor of $\sqrt{R(\kappa)}$ as many errors as it is allowed.

We remark that optimistic decoding is most useful when $d(\kappa)$ and $e(\kappa)$ are well-separated. The Singleton bound says that $d(\kappa) \leq n(1 - R(\kappa)) + 1$, so when $e(\kappa)$ is large (say, close to the Shannon capacity $1 - R(\kappa)$), optimism provides little to no benefit.

Decoding optimizations. The original receiver algorithm R can also be rewritten to be more efficient in practice, by eliminating unnecessary signature verifications. The main observation is that it is unnecessary to verify those triples (m, t, σ) whose value of t is smaller than $t^{\max}$. Of course, the reasoning here is slightly circular, because $t^{\max}$ can only be calculated by checking signatures. However, the receiver can do the following: given the output $\{(m^{(i)}, t^{(i)}, \sigma^{(i)}\}$ of the list decoding algorithm, *sort* the triples by *decreasing value of* $t^{(i)}$. Then verify the triples in this sorted order, and output the first valid $m^{(i)}$.[10]

[10]This description is not entirely equivalent to the previous algorithm, because it does not account for ambiguity (two different but valid messages with the same counter). However, it is easy to see that this new algorithm outputs an incorrect message only when the old algorithm would also do so.

Alternate security reductions. Our proof of Theorem 3.5.1 attains a large forger advantage at the cost of a large circuit to create the forgery. In fact, other reductions are possible which trade forger advantage for circuit size. For example, at the opposite end of this extreme, we can design a reduction that emulates the receiver in *only one* randomly-chosen iteration out of $q(k)$ total; this would reduce the forger's advantage by a factor of $q(k)$, but also decrease the additive term in the circuit size by a factor of $q(k)$.

We can similarly reduce the number of signature verifications done by the reduction at the cost of decreased forger advantage. In fact, we (by necessity) will take this approach in Section 3.7, where we construct coding schemes in a symmetric-key model using message authentication codes.

3.6 Optimality and Necessity of the Model

In our cryptographic construction of a coding scheme, the sender keeps a secret key, maintains some local state between each encoding, and the channel is assumed to be computationally bounded In this section, we show that these features of our construction are actually *essential* for unique decoding under high error rates. Our main tool is an upper bound on the *distance* of error-correcting codes (which shows that plain codes cannot unambiguously decode from high error rates in the worst case). Of course, this on its own is not enough to rule out coding schemes that may rely on randomness, state, and secrecy — but we are able to exploit the bound in constructing adversarial channels that defeat coding schemes that are constrained in certain ways.

To prove all the results of this section, we will be using a generalized form of the Plotkin bound [56] shown by Berlekamp [7]. Let $A_q(n, d)$ denote the maximum size (i.e., number of codewords) of any code over a q-ary alphabet having block length n and minimum distance at least d. Then we have:

Proposition 3.6.1. *For $q, n, d \in \mathbb{N}$, $q \geq 2$:*

$$A_q(n, d) \leq \frac{dq}{dq - n(q-1)}, \qquad \text{if } dq > n(q-1)$$

From this we easily get the following:

Corollary 3.6.2. *Let C be a code over a q-ary alphabet having block length n and distance d. If $|C| \geq n(q-1)+2$, then $d \leq n \cdot \frac{q-1}{q}$.*

In order to exploit Corollary 3.6.2, our proofs will rely on the fact that any coding scheme has sufficiently many distinct messages:

Lemma 3.6.3. *For any coding scheme $\mathcal{CS}$, let $q(k) = |\Sigma_k|$. Then there are at least $n(k) \cdot (q(k)-1)+2$ distinct messages of length k, for all sufficiently large k.*

Proof. There are $q(k)^k$ messages of length k, and $q(k) \geq 2$ for all k. Therefore it suffices to show that $q(k)^{k-1} \geq n(k)+1$ for sufficiently large k. But because $\mathcal{CS}$ is a coding scheme, $n(k) = \text{poly}(k)$, while $q(k)^{k-1}$ is exponential in k. □

3.6.1 Necessity of Local State

Proposition 3.6.4. *Let $\mathcal{CS}$ be any coding scheme in which both the sender and receiver are stateless. Write $q = q(k) = |\Sigma_k|$, $\Sigma = \Sigma_k$, $n = n(k)$, $e = e(k)$ where k is clear from context.*

Then for any $e > n \cdot \frac{q-1}{2q}$,

$$\mathsf{Adv\text{-}CG}^{\mathcal{CS}}_{2,O(n),e}(k) \geq \frac{1}{(n(q-1)+2)^2}$$

for infinitely many k. In particular, if $q(k) = poly(k)$, $\mathcal{CS}$ is not robust under e errors.

In addition, for any *$q(k)$ and for any $e > \frac{n}{2}$, $\mathcal{CS}$ is not robust under e errors.*

These claims hold even if the sender and receiver are randomized and have shared secret or public keys.

Proof. For any $x, x' \in \Sigma^n$, if $\Delta(x, x') \leq n \cdot \frac{q-1}{q}$, define $M(x, x')$ to be some canonical "midpoint" $r \in \Sigma^n$ such that

$$\Delta(x, r) \leq \left\lceil n \cdot \frac{q-1}{2q} \right\rceil \qquad \text{and} \qquad \Delta(r, x') \leq \left\lceil n \cdot \frac{q-1}{2q} \right\rceil.$$

Note that M is computable in $O(n)$ time. We now describe a channel $\mathcal{C}$ whose advantage in the coding game is non-negligible:

1. Choose two *distinct* messages $m, m' \in \Sigma^k$ at random.
2. Query the sender on m, yielding $x \in \Sigma^n$, and pass x to the receiver without error.
3. Query the sender on m', yielding $x' \in \Sigma^n$.
4. If $\Delta(x, x') \leq n \cdot \frac{q-1}{q}$, send r to the receiver, where r is either $M(x, x')$ or $M(x', x)$, each with probability $1/2$ (in either case, $\Delta(r, x') \leq \left\lceil n \cdot \frac{q-1}{2q} \right\rceil \leq e$).

It is clear that this channel makes exactly 2 queries and runs in time $O(n)$.

Now consider an alternate experiment in which a channel queries m' first (yielding x') and m second (yielding x), but Step 4 is syntactically identical. In the following, we condition on the "good" event that $\Delta(x, x') \leq n \cdot \frac{q-1}{q}$. We first observe that the *second received value* r is distributed identically in both experiments — this is because, by the statelessness of the sender, x (likewise x') is distributed identically in both experiments. Then, because the receiver is also stateless, the receiver's output given r is distributed identically in both experiments. The receiver's output is supposed to be different in the two experiments, so in one of the experiments the channel has advantage at least $\frac{1}{2}$ (conditioned on the good event).

It remains to bound $\Pr[\Delta(x, x') \leq n \cdot \frac{q-1}{q}]$. Consider a thought experiment in which $\mathcal{C}$ additionally queries $n(q-1)$ additional random messages (all distinct) before

querying m and m' (sufficiently many unique messages exist by the bound on q^k). The set of all the encodings forms a code of size $n(q-1)+2$, so by Corollary 3.6.2, some pair of encodings will have Hamming distance at most $n \cdot \frac{q-1}{q}$. Because encodings of messages are independent of their order (the sender is stateless), and the set of encoded messages is random, by symmetry we have

$$\Pr[\Delta(x, x') \leq n \cdot \frac{q-1}{q}] \geq \binom{n(q-1)+2}{2}^{-1},$$

which establishes the bound on the channel's advantage.

Now if $q(k) = \text{poly}(k)$, the advantage of the channel is non-negligible in k, so $\mathcal{CS}$ is not robust. Furthermore, for $e > \frac{n}{2}$ (and an alphabet of any size), an argument similar to the one above suffices; we only need that $\Delta(x, x') \leq n$, which happens with certainty. □

3.6.2 Necessity of Sender Secrecy

Proposition 3.6.5. *Let $\mathcal{CS}$ be any coding scheme in which the sender has no secret key (i.e., all its inputs are known to the channel), and use the same notation as in Proposition 3.6.4.*

Then for any $e > n \cdot \frac{q-1}{2q}$,

$$\mathsf{Adv\text{-}CG}^{\mathcal{CS}}_{1,s,e}(k) \geq \frac{1}{n(q-1)+2}$$

for infinitely many k, where $s(k) = O(n \cdot q \cdot \mathsf{time}_S(k))$. In particular, if $q(k) = \text{poly}(k)$, $\mathcal{CS}$ is not robust under e errors.

In addition, for any *$q(k)$ and for $e > \frac{n}{2}$, $\mathcal{CS}$ is not robust under e errors.*

These claims hold even if the sender and receiver are randomized and stateful and the receiver has secret inputs.

Proof. Because the sender is a polynomial-time algorithm with only public inputs, it can be *simulated* by the channel. That is, the channel $\mathcal{C}$ can invoke S as a subroutine. Thus, instead of making actual queries, $\mathcal{C}$ can encode messages itself without invoking the sender. This is a crucial difference: because S is not actually encoding messages, its internal state remains unchanged. The channel can thus simulate the execution of S on a variety of inputs with the same internal state.

Define the midpoint function $M(x, x')$ as in the proof of Proposition 3.6.4. We now describe a channel $\mathcal{C}$ which has good advantage in the coding game:

1. Choose distinct, uniformly-random messages $m_1, \ldots, m_{n(q-1)+2} \in \Sigma^k$. Query the sender on m_1, yielding x_1.

2. Internally *simulate* querying S on messages $m_2, \ldots, m_{n(q-1)+2}$, yielding encodings $x_2, \ldots, x_{n(q-1)+2}$, respectively. (If S is stateful, simulate each encoding with the state of S as it was *when m_1 was queried.*)

3. If there exists a $j \neq 1$ such that $\Delta(x_1, x_j) \leq n \cdot \frac{q-1}{q}$, send r to the receiver, where r is either $M(x_1, x_j)$ or $M(x_j, x_1)$, each with probability $\frac{1}{2}$ (in either case, $\Delta(r, x_1) \leq \left\lceil n \cdot \frac{q-1}{2q} \right\rceil \leq e$).

It is clear that this channel makes exactly 1 query and runs in time $s(k)$.

In the following, we condition on the "good" event that there exists a $j \neq 1$ such that $\Delta(x_1, x_j) \leq n \cdot \frac{q-1}{q}$. In such a case, the value r is identically distributed to one in an alternate experiment in which m_j was the real query, and the encoding of m_1 was simulated — this relies on the fact that all encodings (both real and simulated) use the same state of S. The receiver's output is supposed to be different in the two experiments, but its view is identical in both. Therefore the channel has advantage at least $\frac{1}{2}$ (conditioned on the good event).

It remains to bound the probability of the good event. The set of all encodings $x_1, \ldots, x_{n(q-1)+2}$ forms a code, so by Corollary 3.6.2, there must exist $i \neq j$ such that $\Delta(x_i, x_j) \leq n \cdot \frac{q-1}{q}$. Because the messages are uniform, by symmetry the good event occurs with probability at least $\frac{2}{n(q-1)+2}$, which yields the bound on the channel's advantage.

For the additional claims, the reasoning from the proof of Proposition 3.6.4 applies directly. □

3.6.3 Necessity of a Bounded Channel

Proposition 3.6.6. *Let $\mathcal{CS}$ be any coding scheme in which the receiver is stateless and has only public inputs (i.e., all its inputs are known to the channel), and use the same notation as in Proposition 3.6.4.*

Then for any $e > n \cdot \frac{q-1}{2q}$,

$$\mathsf{Adv\text{-}CG}^{\mathcal{CS}}_{n(q-1)+2,s,e}(k) \geq \frac{1}{2}$$

for infinitely many k, where $s(k) = \exp(n \log q + \mathsf{time}_R(k))$.

In addition, for any *$q(k)$ and for $e > \frac{n}{2}$, $\mathcal{CS}$ is not robust under e errors.*

These claims hold even if the sender and receiver are randomized and stateful and the receiver has secret inputs.

Proof. Note that because the receiver is stateless, its output distribution on a given word r is always the same, regardless of what transmissions have preceded r. Therefore a channel can simulate the behavior of R on many different inputs. We now describe a channel (having large circuit size) which has good advantage in the coding game:

1. Choose arbitrary distinct messages $m_1, \ldots, m_{n(q-1)+2} \in \Sigma^k$.

2. For $j = 1, \ldots, n(q-1)+2$, do the following:

(a) Query the sender on m_j, yielding an encoding $x_j \in \Sigma^n$.

(b) Consider all r such that $\Delta(x_j, r) \leq n \cdot \frac{q-1}{q}$, and compute the output distribution $R(r)$ (this distribution is computable because R is stateless and all its inputs are known to the channel).

If there exists an r such that $\Pr[R(r) \neq m_j] \geq \frac{1}{2}$, corrupt x_j as r, send r to the receiver, and halt. Otherwise, send x_j uncorrupted and loop.

Computing each distribution $R(r)$ can be done in time $\exp(\mathsf{time}_R(k))$ by enumerating over all of R's possible random coins, and there are (very roughly) at most q^n possible choices for r in each iteration. Therefore $\mathcal{C}$ has circuit size $\exp(n \log q + \mathsf{time}_R(k))$.

We now need only argue that a suitable r is eventually found for some query. The encodings $x_1, \ldots, x_{n(q-1)+2}$ form a code, so by Corollary 3.6.2, for some j there exists $i < j$ such that $\Delta(x_i, x_j) \leq n \cdot \frac{q-1}{q}$, so there exists r such that

$$\Delta(x_i, r) \leq \left\lceil n \cdot \frac{q-1}{2q} \right\rceil \qquad \text{and} \qquad \Delta(r, x_j) \leq \left\lceil n \cdot \frac{q-1}{2q} \right\rceil.$$

If the channel did halt on query i (by finding a suitable corrupted word), then $\Pr[R(r) = m_i] \geq \frac{1}{2}$, which means that $\Pr[R(r) \neq m_j] \geq \frac{1}{2}$, as desired.

For the additional claims, the reasoning from the proof of Proposition 3.6.4 applies directly. □

3.7 Symmetric-Key Model and Construction

Our construction from the previous section relied on digital signatures for its cryptographic sieve. In this section, we show that signatures provide just one possible *instantiation* of our broader framework, and that *authenticity* is the central notion that allows us to correct from high error rates. We will show how a stronger setup assumption (specifically, that the sender and receiver share a secret key) and a different cryptographic primitive (namely, message authentication codes) can be used to construct similar coding schemes. Although the qualitative nature of these schemes is the same (even though the setup assumption is stronger), these results are interesting for several reasons:

1. Message authentication codes (MACs) are typically much more efficient than digital signatures in practice (both in terms of algorithmic efficiency and the length of the authenticator tag).

2. MACs can be constructed to be secure against computationally unbounded adversaries (though requiring the receiver to keep state);

3. The alternate constructions illuminate the generality of our decoding framework.

3.7.1 Model and Coding Game

The model of a symmetric-key coding scheme and its interaction with a channel in a coding game is exceedingly similar to the public-key case. Because of these similarities, we do not give an entire formal definition. Instead, we highlight the essential similarities and differences here:

- The key generator algorithm G outputs a *single* secret key sk (not a pair (pk, sk) of keys).
- *Both* the sender algorithm S and receiver algorithm R get sk, but the channel $\mathcal{C}$ does not.
- In the coding game, the channel still mounts an adaptive chosen-message attack and tries to make the receiver output an incorrect decoding.

3.7.2 Construction and Security Reduction

Our construction of a symmetric-key coding scheme is also very similar to the construction in the public-key model. Essentially, we use a MAC (message authentication code) as a "drop-in" replacement for digital signatures. All the other main aspects of the scheme remain the same:

- The sender keeps a local counter or clock t, and uses it to indicate "message freshness" in the same way.
- Although the receiver now has a secret key input, it remains stateless.[11] To decode, the receiver list decodes the received word and chooses the freshest authentic message in the decoded list.
- The channel still does *not* see the output of the receiver at each iteration. Even in the symmetric-key model, this is without loss of generality and does not weaken the model, for the same reasons described in Section 3.4.

The most significant change is in the resulting security reduction, which constructs a MAC forger out of an adversarial channel. The reason is that, because the channel does not know the secret key sk, it cannot detect which decoded triples $(m^{(i)}, t^{(i)}, \sigma^{(i)})$ are authentic. Instead, the reduction follows this alternate strategy:

1. Choose a random iteration $j \in [q(k)]$, where $q(k)$ is the maximum number of queries made by the channel in the coding game.
2. On iterations $j' \neq j$, do nothing with $\mathcal{C}$'s output word $r_{j'}$. On iteration j, get $\mathcal{C}$'s output word r_j, and list-decode it: $\{(m^{(i)}, t^{(i)}, \sigma^{(i)})\} \leftarrow \mathcal{LD}(r_j)$. Choose a random triple (m, t, σ) from this list, and output $((m, t), \sigma)$ as an attempted forgery.

[11] Unless the underlying MAC requires state, as is the case for information-theoretically secure MACs. In any case, the coding scheme imposes no additional state on the receiver.

By reasoning similar to the public-key case, succeeding in the coding game requires the decoded list in *some* iteration to contain a forged triple. By choosing the iteration and triple at random, the advantage of the forger is (at least) a factor of $q(k) \cdot L(\kappa)$ smaller than the advantage of the channel (where $L(\kappa)$ is the size of the decoded list). However, the circuit size of the forger is more reasonable, because it only needs one copy of the $\mathcal{LD}$ algorithm. In effect, this reduction is at the opposite extreme of the size/advantage trade-off discussed in Section 3.5.5.

Chapter 4

Lower Bounds for Fingerprinting Codes

Unlike physical objects, information is easy to duplicate and distribute — especially in the modern age of the Internet.[1] For those in the business of selling information for profit, or those who want to their information to have limited distribution, this is a troubling state of affairs.

In order to prevent buyers from redistributing copies of information products (such as computer games, audio, and video), there have been varied attempts to build technical "copy control" systems. These systems have the goal of making *any* duplication infeasible or very inconvenient. Historically, most such systems have failed in the face of attacks, while simultaneously hindering legitimate uses of the products (such as making personal backup copies, using excerpts, or transferring copies to new hardware).

An alternate approach to *restricting all copying* is to instead reduce the scope of the problem to deal only with *unauthorized redistribution.* In addition to being a potentially easier problem to solve, this has the benefit of allowing legitimate copies. In this chapter, we investigate a particular method for deterring redistribution, called *fingerprinting.*

4.1 Fingerprinting and Watermarking

The central idea behind fingerprinting is that every purchased copy of a work is embedded with extra hidden information via slight differences in the work, and this information is associated to the buyer of that copy by the distributing authority. If one of these copies is later redistributed or resold, it reveals the identity of the user who purchased the copy. This can be used for a variety of purposes, i.e. to subject the user to suspicion and further investigation, or possibly to revoke the user's subscription or access to further services.

[1] In what has become an online mantra, "information wants to be free" (originally attributed to Stewart Brand). Here the word "free" is ambiguous, and can refer either to its price, or to its freedom from secrecy or encumbrance. Usually the latter meaning is the intended one.

The method of embedding hidden information (known as *stegotext*) into another piece of data (known as *covertext*) depends crucially upon the type of the covertext, e.g. computer code, audio, video, written text, etc. The whole class of techniques usually goes under the rubric of *watermarking* (alluding to the "watermark" patterns embedded in official documents like currency, which are only visible under certain lighting conditions), though it is more accurately referred to as *information hiding* or *steganography*. Watermarking has many other applications, such as proving authorship of a work, or preventing devices from using data from unapproved sources. In this work, we will focus only on watermarking as it applies to fingerprinting and deterring redistribution.

In order to be useful for our application, a watermarking scheme needs to have a few essential properties:

1. The marks *should not* be detectable to the user; in particular, the marks should not alter the functionality or perceived qualities of the covertext.

2. The marks *should* be detectable to the distributing authority.

3. The marks *should* be robust to transformations (such as cropping, compressing, etc.) that do not significantly degrade the functionality or perceived qualities of the covertext.

4.1.1 Introducing Coalitions

In our fingerprinting application, there is an additional wrinkle in the use of watermarks: different users may purchase copies of the same work and *collude* by comparing their copies and looking for differences, which reveals the embedded marks. To make matters worse, the colluding users can potentially mix-and-match different parts of their copies, forming a "hybrid" copy whose embedded fingerprint is not directly associated with any of the users.

These issues force us to re-evaluate both our requirements for the watermarking scheme, as well as our hopes for the entire enterprise of fingerprinting. We cannot hope to prevent colluding users from noticing the differences in their copies, nor from producing hybrids that are just as useful as the original copies. Instead, we can relax the requirements on the marking scheme in the following way:

1. Colluding users *should not* be able to detect marks in positions in which all the colluders have the same mark; any hybrid copy produced by the coalition must retain that mark in that position.

2. At a position in which colluders have different marks, a hybrid copy produced by the coalition *may* include any one of the colluders' marks.[2]

[2]This is a relatively strong restriction on the coalition's behavior, but our negative results will work within this model, making them stronger. More relaxed assumptions are also possible, depending on the type of application and strength of the marking scheme: one may allow the colluders to produce a copy with *no mark at all* at a position in which their marks differ. Relaxing even more, one may allow the colluders to produce *any valid mark* (even one not possessed by any of the colluders), or *none at all*.

These requirements are the essence of the *marking assumption*, which is formalized below in Definition 4.2.1.

Under these assumptions, we would like the distributing authority to still be able to trace a hybrid copy to at least one guilty user in the coalition, at least with high probability. (We cannot hope to always identify more than one user, because other users in the coalition may not contribute at all to the final hybrid copy.)

4.1.2 Our Contributions

Most methods of information hiding require a large volume of covertext in order to embed each bit of stegotext robustly. Therefore the *number of marks* in a fingerprint is a critical measure of the complexity of a fingerprinting scheme. In this chapter, we investigate lower bounds on the lengths of fingerprints as a function of the number of colluders the scheme is designed to resist, and the probability of tracing error.

We investigate this problem in a probabilistic model of fingerprinting with binary marks proposed by Boneh and Shaw [13]. By devising and analyzing a probabilistic collusive strategy, we show that the number of marks must be essentially *quadratic* in the number of colluders, times a factor that is *linear* in the log of the inverse probability of error. This lower bound is actually tight, as it matches a subsequent construction of secure fingerprinting codes by Tardos [68].

4.1.3 Prior Work

Fingerprinting was first described in the literature by Wagner [69]. Coalitions in the context of fingerprinting were first considered by Blakley, Meadows, and Purdy [8]. Boneh and Shaw [13] provided a model for collusion, a probabilistic fingerprinting scheme, and a weak lower bound on code length. An alternate approach to deterring redistribution was envisioned by Dwork, Lotspeich, and Naor [23] in the technique of *digital signets*, which embed some private user data in an encryption key that is used to decrypt content. The embedded data is intended to deter users from exposing their keys, because it would compromise their privacy.

Alternate models of marking (using larger alphabets) and collusive attacks (restricting the output to contain only marks from the colluders' copies) are studied under the names of IPP- and TA-codes [16, 65, 64], which admit deterministic constructions and tracing algorithms (but not always efficient ones).

A preliminary version of the work in this chapter appeared in a conference paper by Peikert, Shelat, and Smith [55]. Following its publication, Tardos [68] designed a fingerprinting scheme having optimal length, and an provided an alternate proof of the main result of this chapter.

Notation. For a random variable P and value x, we sometimes use the abbreviation $P_x = \Pr[P = x]$. The support of P is defined as $\sup(P) = \{x : P_x > 0\}$. All logarithms will be base-2, denoted by lg. The *binary entropy function* $H(\cdot)$ is defined to be $H(x) = x \lg \frac{1}{x} + (1-x) \lg \frac{1}{1-x}$ for all $x \in (0,1)$.

For any $s \times t$ matrix M and any subset $S \subseteq [s]$, let $M|_S$ be the $|S| \times t$ matrix which is the restriction of M to rows $i \in S$.

4.2 The Boneh-Shaw Model

Our results apply in the model proposed by Boneh and Shaw [13]. In this model, a fingerprint is a string of symbols over the binary alphabet $\Sigma = \{0, 1\}$, and the goal is to design a *fingerprinting code* (i.e., a set of fingerprints) for which hybrid copies produced by a coalition are traceable to at least one guilty user (assuming the coalition is not too large). It is relatively easy to see that even for three users, two of which may collude, no *deterministic* construction can achieve this goal perfectly (see "Basic Results," below). Boneh and Shaw get around this problem by allowing the code to incorporate *secret randomness*, and permitting the tracing algorithm to have a small probability of *tracing error*.

This model is formalized in the following definition:

Definition 4.2.1 (Fingerprinting scheme, coalitions, strategies)**.** A *fingerprinting scheme* of length m over the alphabet Σ for n users is a distribution $\mathcal{F}$ over pairs (X, σ), where:

- X is an $n \times m$ matrix over the alphabet Σ,
- $\sigma : \Sigma^m \to \mathcal{P}([n])$ is an algorithm which takes a word $w \in \Sigma^m$ and outputs a subset $\sigma(w) \subseteq [n]$ of *accused* users.

For an integer $c \geq 1$, a *c-coalition* is a set $C \subseteq [n]$, $C \neq \emptyset$, with $|C| \leq c$. A *C-strategy* is a (possibly randomized) algorithm ρ that takes the restriction $X|_C$ and outputs a string $y \in \Sigma^m$ subject to the *marking assumption*: for all $j \in [m]$, there must exist an $i \in C$ such that $y_j = X_{i,j}$. For convenience, we will write $\rho(X)$ when it is understood that the input is really $X|_C$.

A fingerprinting scheme is called (c, ϵ)*-secure* if for any c-coalition $C \subseteq [n]$ and any C-strategy ρ, the probability of tracing error

$$\Pr_{X,\sigma,\rho}[\sigma(\rho(X)) = \emptyset \text{ or } \sigma(\rho(X)) \not\subseteq C] \leq \epsilon.$$

Technical remarks on the definition. We do not require the distribution $\mathcal{F}$ or the algorithms ρ and σ to fall within any particular complexity class. However, all coalition strategies that we specify will be efficient. This strengthens the lower bound, because it demonstrates efficient coalitions that can defeat even unbounded tracing algorithms. In addition we can assume that the tracing algorithm σ is deterministic, because its random coins (which are independent of ρ's coins) can be selected in advance by the distribution $\mathcal{F}$.

Without loss of generality we will also assume that σ always accuses exactly one user, i.e. outputs a singleton set. This is because the empty set is always considered

a failure, and accusing more than one user cannot decrease the failure probability. With this simplification, we can write the event of a tracing error more compactly as $\sigma(\rho(X)) \not\subseteq C$.

Basic results. First we recall the argument (from [13]) that no deterministic code (for at least three users) can be secure (even under coalitions of size 2): consider three fingerprints $w, w', w'' \in \{0,1\}^m$ from the code, and let $x \in \{0,1\}^m$ be defined so that x_i is the *majority* symbol of w_i, w'_i, w''_i. Then according to the marking assumption, x can be produced by any coalition of size 2 among $\{w, w', w''\}$. Therefore, the tracing algorithm cannot accuse any single user, because it is possible that the other two users colluded to produce x.

While this reasoning initially appears to hold even for randomized codes, it elides the fact that a coalition may be very *unlikely* to produce such a "majority" word (as long as the code incorporates some secret randomness). In fact, it is exactly this observation that allowed Boneh and Shaw to construct secure fingerprinting codes. They first designed codes of length $O(n^3 \log(n/\epsilon))$ for n users that are $(n-1,\epsilon)$-secure. By concatenation with the codes of [16], Boneh and Shaw also constructed codes for n users having length $O(c^4 \log(1/\epsilon) \log(n/\epsilon))$, which are (c,ϵ)-secure. They also provided a simple argument demonstrating a lower bound of $\Omega(c \log(1/c\epsilon))$ on the length of any (c,ϵ)-secure code.

4.3 Our Lower Bound

We demonstrate a lower bound on code length that is *quadratic* in the number of colluders:

Theorem 4.3.1. *For any $c \geq 2$, and for all sufficiently small $\epsilon > 0$, any (c,ϵ)-secure fingerprinting code for $\geq c+1$ users has length $m \geq C \cdot c^2 \lg(\frac{1}{(c+1)\epsilon})$, where $C > 0$ is an absolute constant (independent of c and ϵ).*

What the bound means. A few remarks on the meaning of our lower bound are in order, mostly concerning the relationship between parameters c and ϵ. First, we stress that the qualifier "for all *sufficiently small* $\epsilon > 0$" is *necessary* for any meaningful bound. Indeed, for $\epsilon > \frac{c}{c+1}$, a trivial tracing algorithm that just accuses a random user (out of a total of $c+1$) will have an (acceptable) error rate $< \epsilon$, no matter how short (or badly-designed) the code is. Furthermore, any lower bound that is demonstrated by analyzing coalitions of size c (out of a total of $c+1$ users) can only apply for $\epsilon < \frac{1}{c+1}$, because the trivial tracing algorithm (applied to those coalitions) is correct with probability $\frac{c}{c+1}$.

Still, one may want to optimize the value of the "sufficiently small" ϵ for which the bound "kicks in." In this respect, the *analysis* we provide here is not the best possible — it only applies for very small values of ϵ (essentially, for $\epsilon < 2^{-2^c}$). Despite this caveat, we believe that the result is still quite meaningful, for several reasons:

1. Most importantly, it implies that no secure fingerprinting code can have length $o(c^2) \cdot \lg \frac{1}{\epsilon}$, which is an important asymptotic barrier.

2. It seems to us that any "natural" construction of a fingerprinting code should be *oblivious* to the relationship between c and ϵ, and therefore its length should be given by a single asymptotic expression in the two parameters. Our bound implies that any such "natural" construction will have length $\Omega(c^2 \lg \frac{1}{\epsilon})$.

3. Without additional context, there seems to be no *a priori* "correct" connection between c and ϵ — they may be completely independent parameters. In certain applications, it may be that only a small number of users (say, 4) can collude, but that we need a very small probability of tracing error (say, 2^{-80}) to overcome "reasonable doubt." For any such application, the analysis we give has very real implications.

We stress that our *coalition strategies* in the proof of Theorem 4.3.1 are, in fact, essentially the best ones possible. A later work by Tardos [68] gave an improved analysis of our coalition strategies using an esoteric statistical notion called *Rényi divergence*, and proved that Theorem 4.3.1 actually "kicks in" for, say, any $\epsilon < 1/c^2$. Additionally, Tardos constructed a fingerprinting code having length $O(c^2 \lg \frac{1}{\epsilon})$, which demonstrates that our lower bound is, in fact, tight. This essentially closes the issue of code length for probabilistic *binary* fingerprinting codes.

4.3.1 Overview of the Argument

In this section we give an overview of the approach we take in proving Theorem 4.3.1. The argument follows this basic skeleton:

1. Choose a collection of coalitions $\{C_j\}$ having no common user: $\cap C_j = \emptyset$.

2. Define C_j-strategies whose output distributions are guaranteed to be "close" in a certain "global" way, *whenever the code is shorter than the bound claimed in Theorem 4.3.1.*

3. Prove that "close" distributions cannot be traced reliably when they are generated by coalitions that have no common user.

4. Conclude that any secure code *must be at least as long as the claimed bound.*

We elaborate on each of these point below.

For Step 1, we choose an arbitrary set of $c+1$ users, say $\{1, \ldots, c+1\}$, and define the c-sized coalitions $C_j = [c+1] \setminus \{j\}$.

For Step 2, the coalition strategies we consider are quite simple: for each (detectable) position in the code, the coalition chooses its output symbol by flipping an independent *biased* coin. The bias is determined by a carefully-chosen function that depends only on the ratio of 1s seen by the coalition at that position of the code. At any position of the code, the coalitions C_j all see almost the same number of 1s: if

users $1, \ldots, c+1$ have a total of i 1s in that position, then coalition $C_j = [c+1] \setminus \{j\}$ sees either i 1s (if user j has a 0) or $i-1$ 1s (if user j has a 1). Therefore for each position of the code, the coalitions use approximately the same bias. As long as the code is short, the distributions on output words all remain close. (As we will explain later, the actual notion of closeness, which we call "min-mass," is somewhat non-standard, but is related to the statistical distance.) Lemmas 4.3.4 and 4.3.7 demonstrate that our coalition strategies do indeed produce close distributions.

For Step 3, suppose (as a thought experiment) that coalitions C_j (having no user in common) generate *identical* output distributions. Because these coalitions have no single user in common, the tracing algorithm cannot reliably accuse any individual user with reasonable confidence. Every user j is equally likely to be innocent, because the coalition C_j (of which user j is *not* a member) is as likely as any other coalition C_i to have created the pirate word. While our strategies do not produce *identical* distributions, similar reasoning still applies when the distributions are close in terms of min-mass; this is shown in Lemma 4.3.3.

4.3.2 Min-Mass of Distributions

Definition 4.3.2. For discrete distributions $P_1, \ldots, P_t$ defined over a set S, define the *min-mass* $\mathsf{mm}(P_1, \ldots, P_t)$ to be:

$$\mathsf{mm}(P_1, \ldots, P_t) = \sum_{x \in S} \min_{i \in [t]} \{\Pr[P_i = x]\}.$$

It is apparent that the min-mass of any distributions lies in $[0, 1]$: distributions with disjoint support have min-mass 0, while identical distributions have min-mass 1. The *larger* the min-mass, the *closer* we consider the distributions to be.

Min-mass is related to the more standard notion of statistical distance between two distributions. In fact, the notions are equivalent for *pairs* of distributions: $\mathsf{mm}(P_1, P_2) = 1 - SD(P_1, P_2)$. However, for more than two distributions the min-mass is a better notion of "global" closeness. For example, it is easy to construct three distributions which are pairwise close (i.e., the statistical distance between any pair is small) but which have zero min-mass.

4.3.3 Relating Min-Mass to Tracing Error

We now connect the min-mass of coalition distributions with the error probability of a fingerprinting scheme. The error probability is essentially *at least as large* as the min-mass of any distributions generated by coalitions having no user in common.

Lemma 4.3.3. *Let $\{C_j\}_{j \in [t]}$ be a collection of c-coalitions with no common user (i.e., $\cap_{j \in [t]} C_j = \emptyset$) for a fingerprinting scheme $\mathcal{F}$, and let ρ_j be a C_j-strategy for $j \in [t]$. If $\mathcal{F}$ is an (ϵ, c)-secure fingerprinting scheme, then:*

$$t \cdot \epsilon \geq E_{X,\sigma}\left[\mathsf{mm}(\rho_1(X), \ldots, \rho_t(X))\right].$$

Proof. We will first bound the quantity inside the expectation for any fixed X, σ. The lemma follows by a standard averaging argument.

Consider any fixed X, σ from $\mathcal{F}$. Let $\epsilon_{X,\sigma}$ be an upper bound on the failure probability of the tracing algorithm for all c-coalitions C and C-strategies ρ:

$$\epsilon_{X,\sigma} \geq \Pr_{\rho}[\sigma(\rho(X)) \not\subseteq C].$$

We will write $\epsilon' = \epsilon_{X,\sigma}$ when X, σ are clear from context.

Now for every $j \in [t]$ we have

$$\epsilon' \geq \Pr_{\rho_j}[\sigma(\rho_j(X)) \not\subseteq C_j] = \sum_{w \in \Sigma^m} \Pr[\sigma(w) \not\subseteq C_j] \cdot \Pr[\rho_j(X) = w],$$

where $\sigma(w) \not\subseteq C_j$ is actually a deterministic event (because σ is deterministic), and is independent from the event $\rho_j(X) = w$ (which depends only on the coins of ρ_j).

By summing over all $j \in [t]$, we have

$$\begin{aligned} t \cdot \epsilon' &\geq \sum_{j \in [t]} \sum_{w \in \Sigma^m} \Pr[\sigma(w) \not\subseteq C_j] \cdot \Pr[\rho_j(X) = w] \\ &\geq \sum_{w \in \Sigma^m} \left(\sum_{j \in [t]} \Pr[\sigma(w) \not\subseteq C_j] \right) \cdot \min_{i \in [t]} \Pr[\rho_i(X) = w] \\ &\geq 1 \cdot \mathsf{mm}(\rho_1(X), \ldots, \rho_t(X)) \end{aligned}$$

where we have used the fact that there exists some $j \in [t]$ such that $\sigma(w) \not\subseteq C_j$: for if not, $\sigma(w) \subseteq \cap\, C_j = \emptyset$, but σ always outputs a singleton set.

We complete the proof by averaging over X, σ, observing that $\epsilon \geq E_{X,\sigma}[\epsilon_{X,\sigma}]$. □

4.3.4 The Coalitions and Their Strategies

Here we formally specify the coalition strategies that will be close in terms of min-mass. The choice of coalitions is somewhat arbitrary. For simplicity, we let $C = [c+1]$ and define $C_j = C \setminus \{j\}$ for each $j \in [c+1]$. We remark that, as required by Lemma 4.3.3, the C_j are disjoint: $\cap_{j \in [c+1]} C_j = \emptyset$.

The strategy ρ_j for C_j is as follows: for each position i of the code, the coalition counts the ratio r_i of 1s it sees at position i, i.e. the number of 1s divided by c. The coalition then outputs at position i the result of a *biased coin flip*, i.e. a sample drawn independently from a distribution P_i where $\Pr[P_i = 1] = b(r_i)$, and b is a *bias function* described below.

The bias function b is defined over the interval $[0, 1]$ as

$$b(r) = 3r^2 - 2r^3.$$

A graph of the bias function is shown in Figure 4-1. It is easily verified that

1. $b([0,1]) = [0,1]$,

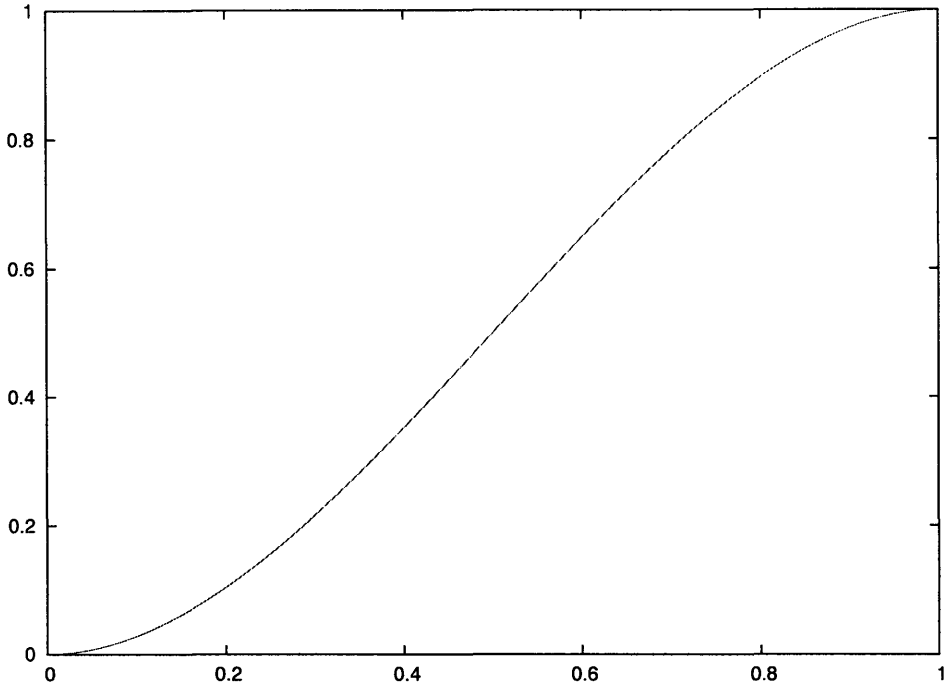

Figure 4-1: Graph of the bias function $b(\cdot)$

2. $b(0) = 0$ and $b(1) = 1$, and

3. $1 - b(r) = b(1 - r)$, i.e. b is symmetric.

Property 1 of the bias function guarantees that the coalition strategies use well-defined probabilities. Property 2 ensures that the coalitions obey the Marking Assumption: if a coalition sees only symbol s at some position, then it outputs s at that position with certainty. Property 3 means that the coalitions are not sensitive to the *representations* of the symbols 0 and 1 as embedded in a document. That is, the behavior of a coalition is the same, whether it regards a particular kind of mark as a 0 or as a 1. The only relevant quantity is the *ratio* of occurrences of one mark to the other at each position.

As for the specific the function $b(\cdot)$, it is chosen so that the function grows slowly (i.e., as a parabola) near the endpoints of $[0, 1]$, but more quickly (i.e., linearly) through the middle. This is because biases near 0 and 1 are easier to distinguish than biases around $\frac{1}{2}$. (See Section 4.3.5 below for a more precise explanation of the choice of our bias function.)

In order to analyze these coalition strategies, we will need to calculate the min-mass of their induced distributions. We now show how this min-mass can be simplified

into an expression involving only the differences between pairs of Bernoulli processes.

First we define some notation: for $i \in [c]$ and an integer $n \geq 0$, let $B_{j,c,n}$ be the (Bernoulli) distribution over Σ^n induced by n independent trials $X_1, \ldots, X_n$ where $\Pr[X_k = 1] = b(\frac{j}{c})$. (Abusing notation slightly, we will also write $B_{j,c,n}$ to denote a random variable having that distribution.)

Lemma 4.3.4. *Let X be any fixed code of length m with at least $c+1$ rows, and let X' be the restriction of X to its first $c+1$ rows. For each "column type" $t \in \Sigma^{c+1}$, let M_t be the set of indices i for which the ith column of X' is t, and let $m_t = |M_t|$.*

Then for the coalition strategies ρ_j described above,

$$\mathsf{mm}(\rho_1(X), \ldots, \rho_{c+1}(X)) \geq \prod_{\substack{t \in \Sigma^{c+1} \\ wt(t) \in [c]}} \mathsf{mm}(B_{wt(t),c,m_t}, B_{wt(t)-1,c,m_t}).$$

Proof. First, we observe that for the all-0s (respectively, all-1s) columns of X', every coalition outputs 0 (resp., 1) with certainty. Therefore the min-mass only depends on the behavior of the coalitions on the remaining columns, i.e. those having type t with $\mathrm{wt}(t) \in [c]$.

For any word $x \in \Sigma^m$ and column type $t \in \Sigma^{c+1}$, define $x^{(t)} \in \Sigma^{m_t}$ to be the restriction of x to those indices in M_t. Then we have:

$$\begin{aligned}
&\mathsf{mm}(\rho_1(X), \ldots, \rho_{c+1}(X)) \\
&= \sum_{w \in \Sigma^m} \min_j \left\{ \prod_{\substack{t \in \Sigma^{c+1} \\ \mathrm{wt}(t) \in [c]}} \Pr[\rho_j(X)^{(t)} = w^{(t)}] \right\} \\
&\geq \sum_{w \in \Sigma^m} \prod_t \min_j \left\{ \Pr[\rho_j(X)^{(t)} = w^{(t)}] \right\} \\
&= \sum_{w \in \Sigma^m} \prod_t \min \left\{ \Pr[B_{\mathrm{wt}(t),c,m_t} = w^{(t)}], \Pr[B_{\mathrm{wt}(t)-1,c,m_t} = w^{(t)}] \right\} \qquad (4.1) \\
&= \prod_t \sum_{w \in \Sigma^{m_t}} \min \left\{ \Pr[B_{\mathrm{wt}(t),c,m_t} = w], \Pr[B_{\mathrm{wt}(t)-1,c,m_t} = w] \right\} \qquad (4.2) \\
&= \prod_t \mathsf{mm}(B_{\mathrm{wt}(t),c,m_t}, B_{\mathrm{wt}(t)-1,c,m_t}).
\end{aligned}$$

Equation (4.1) follows from the fact that the number of 1s a coalition sees is the same on *all* columns of M_t, and is either $\mathrm{wt}(t) - 1$ or $\mathrm{wt}(t)$. Hence every coalition's output distribution on those columns is either $B_{\mathrm{wt}(t),c,m_t}$ or $B_{\mathrm{wt}(t)-1,c,m_t}$. Equation (4.2) follows by decomposing w into the independent parts $w^{(t)}$, and rearranging the sum and product. □

4.3.5 Min-Mass of Binomial Distributions

Intuition. It remains to analyze the quantities $\mathsf{mm}(B_{j,c,n}, B_{j-1,c,n})$, that is, the similarity between a coin having bias $b(\frac{j}{c})$ and one having bias $b(\frac{j-1}{c})$, measured by flipping the coins n times. As we have explained above, for *pairs* of distributions the function mm equals (twice) the error probability of an optimal distinguisher between the two distributions. One way of measuring the ability to distinguish between samples from two distributions is by the *(Kullback-Leibler) divergence* [45] between the distributions (defined below). A result known as Stein's Lemma [17, Section 12.8] says that the error rate of an optimal distinguisher decreases exponentially with the divergence multiplied by the number of trials. That is, if two coins have divergence d, then the optimal distinguisher between n trials of one coin or the other has error rate $\exp(-\Theta(dn))$.

For coins having bias bounded away from 0 and 1, the divergence is approximately the *square* of the difference between the biases. That is, the divergence between bias $\frac{1}{2}$ and $\frac{1}{2}+\epsilon$ is $\Theta(\epsilon^2)$. Therefore one needs about $\Theta(\epsilon^{-2})$ trials to distinguish between these coins. For coins having bias near 0 or 1, however, the situation is different: the divergence between bias 0 and bias ϵ is only $\Theta(\epsilon)$, therefore only $\Theta(\epsilon^{-1})$ trials are needed.

These facts motivate the "shape" of our bias function, which grows slowly at the endpoints but more quickly through the center. The shape is such that the divergence between adjacent biases $b(\frac{j-1}{c})$ and $b(\frac{j}{c})$ is $O(\frac{1}{c^2})$ for all j. Intuitively, this is the source of our c^2 lower bound: each position of the code contributes only $O(\frac{1}{c^2})$ to the overall divergence, so to discriminate with an ϵ probability of error, at least $\Omega(c^2 \log \frac{1}{\epsilon})$ positions should be required.

Unfortunately, this intuition does not immediately translate into a rigorous proof. One reason is that Stein's Lemma is only a loose *asymptotic* result, as it depends on the Law of Large Numbers. The bound on the error rate only applies for a sufficiently large (and unspecified) number of trials, and in our setting we have no guarantee that the number of columns of each type will be sufficiently large. Additionally, the asymptotics in the statement of Stein's Lemma leave out lower-order terms that we cannot ignore in our analysis. Therefore we must perform a more precise accounting of the min-mass between binomial distributions.

Precise analysis. While we will not make use of Stein's Lemma, the Kullback-Leibler divergence will still emerge in our analysis. We define it and state a few of its properties here:

Definition 4.3.5 (Kullback-Leibler divergence [45])**.** For discrete random variables P, Q, the *Kullback-Leibler divergence* (also known as *relative entropy*) is defined as:

$$\mathsf{KL}(P\|Q) = \sum_{x \in \sup(P)} P_x \lg \frac{P_x}{Q_x}.$$

The divergence is defined only if $\sup(P) \subseteq \sup(Q)$. For binary variables P, Q for which $P_1 = p$ and $Q_1 = q$, we write $\mathsf{KL}_{p,q} = \mathsf{KL}(P\|Q)$.

Fact 4.3.6. *For $p, q \in (0,1)$,*

$$\mathsf{KL}_{p,q} \leq \frac{1}{\ln 2} \cdot \frac{(p-q)^2}{q(1-q)}$$

Proof. We use the fact that $\lg(1+x) \leq \frac{x}{\ln 2}$ for any $x > -1$. The result follows by basic manipulations:

$$\begin{aligned}
\mathsf{KL}_{p,q} &= p \lg \frac{p}{q} + (1-p) \lg \frac{1-p}{1-q} \\
&\leq \frac{p}{\ln 2} \cdot \frac{p-q}{q} + \frac{1-p}{\ln 2} \cdot \frac{p-q}{1-q} \\
&= \frac{1}{\ln 2} \cdot \frac{(p-q)^2}{q(1-q)}. \quad \square
\end{aligned}$$

Lemma 4.3.7. *For any integer $c \geq 2$, any $j \in [c]$, and any integer $n \geq 0$,*

$$\mathsf{mm}(B_{j,c,n}, B_{j-1,c,n}) \geq 2^{-Kn/c^2} \cdot \frac{1}{4(n+1)^2},$$

where $K > 0$ is an absolute constant (independent of n and c).

Proof. First we dispense with the "endpoint" values of $j = 1$ and $j = c$. If $j = 1$, then the biases of the two coins are $p = 0$ and $q = b(\frac{1}{c}) = \frac{3}{c^2} - \frac{2}{c^3} \leq \frac{1}{2}$. The support of the first coin is simply the word $w = 0^n$, which occurs with probability 1. So the min-mass is $\Pr[B_{j,c,n} = 0^n] = (1-q)^n$, and

$$\lg(1-q)^n = -n \lg \left(1 + \frac{q}{1-q}\right) \geq -n \cdot \frac{2q}{\ln 2} \geq -K \cdot \frac{n}{c^2}$$

for some constant K. A symmetric analysis applies for the case $j = c$.

Now suppose $j \in \{2, \ldots, c-1\}$. For the remainder of the proof, let $p = b(\frac{j-1}{c})$ and $q = b(\frac{j}{c}) > p$, the biases of the two coins we are considering, and let $\delta = q - p \leq \frac{1}{2}$. We divide the analysis into two separate cases: $n < 1/\delta$ ("small" n) and $n \geq 1/\delta$ ("large" n).

First we consider the case $n < 1/\delta$. To bound the min-mass, we bound the probability of each $w \in \{0,1\}^n$ under both $B_{j,c,n}$ and $B_{j-1,c,n}$ by taking the product of the least likelihoods for each of the coordinates of w:

$$\begin{aligned}
\mathsf{mm}(B_{j,c,n}, B_{j-1,c,n}) &\geq \sum_{i=0}^{n} \binom{n}{i} p^i (1-q)^{n-i} \\
&= (p+1-q)^n \\
&= (1-\delta)^n \\
&\geq (1-\delta)^{1/\delta}.
\end{aligned}$$

Now because $\delta \in (0, \frac{1}{2}]$ and $(1-\delta)^{1/\delta}$ is strictly decreasing, the min-mass is $\geq \frac{1}{4}$.

Now we consider the case $n \geq 1/\delta$. Then the interval $[np, nq]$ has length ≥ 1, so it must contain an integer $s = rn$ for some $r \in [p, q]$. To bound the min-mass, we bound the probability that $B_{j-1,c,n}$ takes a value w having weight exactly $s = rn$. Observe that for any such w,

$$\Pr[B_{j-1,c,n} = w] = p^{rn}(1-p)^{(1-r)n}.$$

Therefore

$$\begin{aligned}\Pr[\mathrm{wt}(B_{j-1,c,n}) = rn] &= \binom{n}{rn} p^{rn}(1-p)^{(1-r)n} \\ &\geq (n+1)^{-2} 2^{nH(r)} p^{rn}(1-p)^{(1-r)n}\end{aligned}$$

by the absolute lower bound on binomial coefficients $\binom{n}{\beta n} \geq (n+1)^{-2}2^{nH(\beta)}$ [19], where $H(\cdot)$ is the binary entropy function. Taking logarithms and rearranging, we have:

$$\begin{aligned}\lg \Pr[\mathrm{wt}(B_{j-1,c,n}) = rn] + 2\lg(n+1) &\geq n\left(H(r) + r\lg p + (1-r)\lg(1-p)\right) \\ &= n\left(r\lg\frac{p}{r} + (1-r)\lg\frac{1-p}{1-r}\right) \\ &= -n \cdot \mathsf{KL}_{r,p}\end{aligned}$$

Now by Fact 4.3.6,

$$\mathsf{KL}_{r,p} \leq \frac{1}{\ln 2} \cdot \frac{(r-p)^2}{p(1-p)} \leq \frac{1}{\ln 2} \cdot \frac{(q-p)^2}{p(1-p)}.$$

By definition of the function $b(\cdot)$, one can easily verify that $\frac{(q-p)^2}{p(1-p)} \leq K' \cdot \frac{1}{c^2}$ for some constant $K' > 0$. It follows that

$$\lg \Pr[\mathrm{wt}(B_{j-1,c,n}) = rn] \geq -Kn/c^2 - 2\lg(n+1)$$

for some constant $K > 0$. A virtually identical analysis applies for the distribution $B_{j,c,n}$ as well. Because every w having weight rn is equally likely under $B_{j-1,c,n}$ (and $B_{j,c,n}$) the bound on the min-mass follows. □

4.3.6 Proof of Theorem

We now have all the tools needed to demonstrate the lower bound stated in Theorem 4.3.1.

Our first step is to bound from below the expectation (over X, σ) of the min-mass of the coalition strategies. In fact, we will bound the min-mass of the strategies for an *arbitrary* fixed X, which provides a bound on the expectation as well. By

Lemma 4.3.4,

$$\begin{aligned}
\mathsf{mm}(\rho_1(X), \ldots, \rho_{c+1}(X)) &\geq \prod_{\substack{t \in \Sigma^{c+1} \\ \mathrm{wt}(t) \in [c]}} \mathsf{mm}(B_{\mathrm{wt}(t),c,m_t}, B_{\mathrm{wt}(t)-1,c,m_t}) \\
&\geq \prod_t 2^{-Km_t/c^2} \cdot (2m_t + 2)^{-2} \\
&= 2^{-Km/c^2} \cdot \prod_t (2m_t + 2)^{-2} \\
&\geq 2^{-Km/c^2} \cdot \left(\frac{2m}{2^c} + 2\right)^{-2^{c+2}}
\end{aligned}$$

where the last inequality comes from the fact that $m = \sum_t m_t$, and the expression is minimized when all the $(2^{c+1} - 2)$ values of m_t are equal (and hence, $m_t \leq m/2^c$).

Now by assumption, the fingerprinting code is (ϵ, c)-secure, so by Lemma 4.3.3,

$$(c+1)\epsilon \geq 2^{-Km/c^2} \cdot \left(\frac{2m}{2^c} + 2\right)^{-2^{c+2}}.$$

This implies that either

$$\sqrt{(c+1)\epsilon} \geq 2^{-Km/c^2} \qquad \text{or} \qquad \sqrt{(c+1)\epsilon} \geq \left(\frac{2m}{2^c} + 2\right)^{-2^{c+2}}$$

(or both). In the first case, a few easy manipulations yield:

$$m \geq \frac{1}{2K} \cdot c^2 \lg \frac{1}{(c+1)\epsilon}.$$

In the second case, we have:

$$m \geq 2^{c-1} \left[\left(\frac{1}{(c+1)\epsilon}\right)^{1/2^{c+3}} - 2 \right].$$

However, for all sufficiently small $\epsilon > 0$, this bound exceeds the previous one.[3] Therefore, we have that for sufficiently small ϵ, $m \geq C \cdot c^2 \lg(\frac{1}{(c+1)\epsilon})$, as desired.

[3]Specifically, for $\lg \frac{1}{(c+1)\epsilon} \geq 2^{c+3} \cdot 6c$, the second bound is worse than the first.

Bibliography

[1] Adi Akavia, Shafi Goldwasser, and Shmuel Safra. Proving hard-core predicates using list decoding. In *FOCS*, pages 146–. IEEE Computer Society, 2003.

[2] Werner Alexi, Benny Chor, Oded Goldreich, and Claus-Peter Schnorr. Rsa and rabin functions: Certain parts are as hard as the whole. *SIAM J. Comput.*, 17(2):194–209, 1988.

[3] Noga Alon and Joel H. Spencer. *The Probabilistic Method.* Wiley-Interscience, New York, second edition, August 2000.

[4] Mihir Bellare and Silvio Micali. How to sign given any trapdoor permutation. *J. ACM*, 39(1):214–233, 1992.

[5] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, pages 92–111, 1994.

[6] Elwyn Berlekamp and Lloyd Welch. Error correction of algebraic block codes. US Patent Number 4,633,470, 1986.

[7] Elwyn R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill, New York, 1968.

[8] G. R. Blakley, Catherine Meadows, and George B. Purdy. Fingerprinting long forgiving messages. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 180–189. Springer, 1985.

[9] Manuel Blum. Coin flipping by telephone. In *CRYPTO*, pages 11–15, 1981.

[10] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[11] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[12] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[13] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.

[14] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/`.

[15] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[16] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

[17] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley and Sons, Inc., 1991.

[18] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[19] I. Csiszár and J. Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems.* Academic Press, New York, 1982.

[20] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[21] Yan Ding, Parikshit Gopalan, and Richard J. Lipton. Error correction against computationally bounded adversaries. *Theory of Computing Systems*, 2006. To appear.

[22] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552. ACM, 1991.

[23] Cynthia Dwork, Jeffrey B. Lotspiech, and Moni Naor. Digital signets: Self-enforcing protection of digital information (preliminary version). In *STOC*, pages 489–498, 1996.

[24] Cynthia Dwork, Ronen Shaltiel, Adam Smith, and Luca Trevisan. List-decoding of linear functions and analysis of a two-round zero-knowledge argument. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 101–120. Springer, 2004.

[25] Peter Elias. List decoding for noisy channels. In *Wescon Convention Record*, Part 2, Institute of Radio Engineers (now IEEE), pages 94–104, 1957.

[26] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Odlyzko [54], pages 186–194.

[27] Michel X. Goemans, editor. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA.* ACM Press, 2003.

[28] Oded Goldreich. On security preserving reductions — revised terminology. Technical Report MCS00-03, Weizmann Institute Of Science, 2000.

[29] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[30] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In Johnson [42], pages 25–32.

[31] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[32] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[33] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[34] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[35] Venkatesan Guruswami. List decoding with side information. In *IEEE Conference on Computational Complexity*, pages 300–. IEEE Computer Society, 2003.

[36] Venkatesan Guruswami, Johan Håstad, Madhu Sudan, and David Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48(5):1021–1034, 2002.

[37] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In Goemans [27], pages 126–135.

[38] Venkatesan Guruswami and Atri Rudra. Explicit capacity-achieving list-decodable codes. In *Proceedings of the 38th ACM Symposium on Theory of Computation (STOC)*, May 2006.

[39] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[40] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. In *STOC*, pages 181–190, 2000.

[41] Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950.

[42] D. S. Johnson, editor. *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, 15-17 May 1989, Seattle, Washington, USA*. ACM, 1989.

[43] Burton S. Kaliski Jr. A pseudo-random bit generator based on elliptic logarithms. In Odlyzko [54], pages 84–103.

[44] Akinori Kawachi and Tomoyuki Yamakami. Quantum hardcore functions by complexity-theoretical quantum list decoding. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2006.

[45] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

[46] Michael Langberg. Private codes or succinct random codes that are (almost) perfect. In *FOCS*, pages 325–334. IEEE Computer Society, 2004.

[47] Richard J. Lipton. A new approach to information theory. In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, *STACS*, volume 775 of *Lecture Notes in Computer Science*, pages 699–708. Springer, 1994.

[48] Michael Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, 1996.

[49] Anna Lysyanskaya, Roberto Tamassia, and Nikos Triandopoulos. Multicast authentication in fully adversarial networks. In *IEEE Symposium on Security and Privacy*, pages 241–. IEEE Computer Society, 2004.

[50] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Comm. ACM*, 24(9):583–584, 1981.

[51] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction against computationally bounded noise. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[52] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In Johnson [42], pages 33–43.

[53] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. ACM, 1990.

[54] Andrew M. Odlyzko, editor. *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science.* Springer, 1987.

[55] Chris Peikert, Abhi Shelat, and Adam Smith. Lower bounds for collusion-secure fingerprinting. In *SODA*, pages 472–479, 2003.

[56] Morris Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6:445–450, September 1960.

[57] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8(2):300–304, June 1960.

[58] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[59] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM Press, 1990.

[60] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[61] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July 1948.

[62] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[63] Mohammad Amin Shokrollahi and Hal Wasserman. Decoding algebraic-geometric codes beyond the error-correction bound. In *STOC*, pages 241–248, 1998.

[64] Alice Silverberg, Jessica Staddon, and Judy L. Walker. Efficient traitor tracing algorithms using list decoding. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2001.

[65] Jessica Staddon, Douglas R. Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Transactions on Information Theory*, 47(3):1042–1049, 2001.

[66] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.

[67] Madhu Sudan. List decoding: Algorithms and applications. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *IFIP TCS*, volume 1872 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2000.

[68] Gábor Tardos. Optimal probabilistic fingerprint codes. In Goemans [27], pages 116–125.

[69] Neal R. Wagner. Fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 18–22, 1983.

[70] J. M. Wozencraft. List decoding. *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, 48:90–95, 1958.

[71] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91. IEEE, 1982.