Efficient Cryptographic Tools for Secure Distributed Computing

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Aleksandr Yampolskiy

Dissertation Director: James Aspnes

May 2006

To my parents, Irina Lagutina and Vladimir Yampolskiy.

Abstract

Efficient Cryptographic Tools for Secure Distributed Computing

Aleksandr Yampolskiy

2006

This thesis presents a set of efficient and practical tools that can be used to enhance security of existing distributed systems. Our tools achieve marked improvements in bit, round, and communication complexity compared to generic multi-party solutions.

We begin by introducing a new cryptographic primitive, called the **blind coupon mechanism** (BCM). The BCM is effectively an AND-homomorphic bit commitment scheme. It provides a missing link in protocol design because it was not known how to construct such commitments before.

Next, we give a simple construction of a **verifiable random function** (VRF). A VRF is a pseudo-random function that provides publically verifiable proofs of its outputs' correctness. Our VRF is quite practical compared to prior constructions. It has already been used to construct a compact e-cash scheme [CHL05] and a distributed electronic lottery [CHYC05].

We then use our VRF to construct an $O(1)$ round distributed protocol for the Luby-Rackoff construction, yielding the first reasonably efficient threshold **pseudo-random permutation** (PRP). Many protocols that use PRPs like a CBC block cipher mode can now be translated into the distributed setting.

Finally, we consider incentive compatibility in the context of distributed systems. We propose a simple game for modeling containment of the spread of viruses in a network of users, and show that allowing selfish users decide whether or not to install a security technology can be highly undesirable.

Acknowledgements

My experience as a graduate student at Yale has been amazing, and there are many people to thank for that.

I want to first thank my advisor, James Aspnes, for everything he has done for me. From the start, Jim has taught me that doing research should be fun. He gave me the freedom to explore problems, which I found interesting, even though they seemed implausible at times. He was always patient and encouraging even as I called him up on Sundays to vent about my proofs being false. Jim's optimism and sense of humor kept me going throughout graduate school.

Yevgeniy Dodis has been like a second advisor to me. He was kind enough to agree to meet me back when I was a clueless first-year Ph.D. student. Since then, he has always found time to talk to me despite his impending STOC or FOCS deadlines. Yevgeniy was a constant source for brilliant ideas, and I cannot thank him enough for what he has done.

Joan Feigenbaum convinced me to come to Yale for graduate school, for which I am indebted to her. She has closely followed my academic progress and kept me on the right track with her friendly advice. My desire to justify Joan's confidence greatly motivated me to finish up the dissertation.

Richard Yang has always found time to meet with me despite his hectic schedule. I sincerely thank him for all his help and for agreeing to serve on my committee.

Many results in this dissertation are joint work with my co-authors: Jim, Joan, Yevgeniy, Kevin Chang, Zoë Diamadi, Kristian Gjøsteen, René Peralta, Moti Yung, and Sheng

Zhong. It was truly a privilege to collaborate with every one of them. Their hard work molded this dissertation into what it is today, and for that I thank them.

Throughout my graduate studies, I learnt much from the faculty members of the department. Particularly, I would like to thank Dana Angluin and Michael Fischer for their insightful questions and suggestions about my research.

I thank Linda Dobb and other administrative staff for taking care of numerous university formalities.

Thanks are also due to my fellow graduate students, who provided me with much needed distraction from arduous work. I want to especially thank (in no particular order) Hong Jiang, Jatin Shah, Gauri Shah, Ronny Dakdouk, Felipe Saint-Jean, David Goldenberg, Vijay Ramachandran, Adam Poswolsky, and Yitong Yin for their friendship.

But, most of all, I would like to thank my parents, Irina Lagutina and Vladimir Yampolskiy, who have always believed in my success. They gave up many of their dreams so that I could achieve mine. It is to them that I dedicate this dissertation.

Aleksandr Yampolskiy
New Haven, CT, USA
February 2006

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."

-Leslie Lamport

Much of the research in computer security has occurred in the context of **centralized systems**. Centralized systems were state-of-the-art in the 1970's. They consign all functionality and information to a single server accessed by multiple users. A prime example of this is the client-server paradigm, which is used by many databases and Web servers. The centralized system's owner has complete control over its security aspects. In fact, it is relatively easy to secure a centralized system: We can place the central server in a locked room, allow only authorized users to log in, and protect the operating system from user processes. With all "security eggs" in one basket, users know who to speak to when problems arise. They can decide whether or not to trust the server's owner with their data.

Centralized systems may be attractive from a security standpoint because only one server needs to be protected. This is also their drawback: If the server goes down, the entire system stops functioning. Recent years have seen a proliferation of **distributed systems** that harness the power and resources of multiple servers communicating over the network to accomplish useful tasks. In the most general case, any computer can join the network and instantly begin participating in the computation. Unlike centralized systems, distributed systems do not fail all at once. At the same time, it may be much more difficult to secure distributed systems. They require a different set of tools and techniques than those required by centralized systems. Much of the equipment may be located in insecure locations. It thus may become impossible to prevent corruption of some of the servers. The network can be tapped at any point and corrupt servers with modified operating systems can eavesdrop on network traffic and insert bad data into the system. As a result, the data circulating around the network is no longer authoritative.

In the centralized setting, the host is essentially a black box, connected to a user by a fixed link, that either works correctly or it does not. In the distributed system, this fundamental assumption is no longer valid because the host is distributed. The techniques for solving these problems involve the use of cryptography. Because securing a distributed system is too complex a problem to tackle directly, the problem is broken into smaller manageable pieces. Then a cryptographic primitive is introduced for each of the pieces. Primitives' implementations may vary: they may be impractical or quite efficient and they may be secure based on certain assumptions or on existence of other primitives. The primitives can also be combined (albeit not without some difficulty) to produce secure distributed protocols.

1.1 Our Contributions

We propose a set of new cryptographic tools that can be used to enhance security of distributed systems—the **blind coupon mechanism**, the **verifiable random function** with short proofs and keys, and the **threshold pseudo-random permutation**. The blind coupon mechanism allows parties in the network to communicate while keeping even the fact of their communication secret. The verifiable random function allows parties to flip coins (a necessary ingredient in steps of many distributed protocols), while ensuring coin flips' correctness. The threshold pseudo-random permutation allows parties holding shares of the secret key to jointly encrypt and decrypt circulating messages. The common thread is that we provide solutions to carefully stated problems, and we set out to provide them more efficiently or based on fewer or weaker assumptions than before. By tailoring our solutions to specific problems, we achieve marked improvements in efficiency compared to generic multi-party solutions [BoGW88]. Our tools provide important missing links in protocol design. They are efficient, secure, and can be used either in standalone applications or as building blocks in designing secure distributed systems.

In addition to proposing new tools, we also examine security of distributed systems from combined economic and cryptographic perspectives. Security is very much dependent on the behavior of individual servers. In theory, honest servers can defend themselves from disruptive behavior of malicious servers by using cryptographic tools similar to the ones we proposed. In practice, the situation is different. Distributed systems may have multiple points of security control with contradictory interests. While one server's owner may be eager to adopt a security mechanism, another server's owner may not, preferring an insecure system to a relatively secure one because of performance concerns or other considerations. [1] To address these issues,

[1]To illustrate, a user with a dial-up connection may hesitate to download 25 megabytes of

we propose an economic model for security of distributed systems, where each server must choose to either install the security technology or risk possible compromise of his system.

We now describe our tools and security model in more detail.

1.2 Blind Coupon Mechanism

Our first tool is a new cryptographic primitive called the **blind coupon mechanism** (BCM). The BCM consists of two sets of **dummy coupons**, D_{SK}, and **signal coupons**, S_{SK}. An attacker, who lacks the secret key SK, cannot distinguish between these two sets. He also cannot fabricate new signal coupons on his own. The BCM allows all users (even the ones lacking SK) to **combine** coupons such that a combination of a signal coupon with any other coupon yields a signal coupon, while a combination of dummy with dummy yields a dummy coupon.

The BCM has natural and important applications. If signal coupons are used to encode a "0" and dummy coupons a "1", then a BCM can be viewed as a special AND-homomorphic bit commitment. As far as we know, it has not been known how to construct such commitments before. The BCM thus provides a missing link in protocol design. Using BCM together with techniques of Brassard *et al.* [BCC88], we can obtain short non-interactive proofs of circuit satisfiability, whose length is linear in the number of AND gates in the circuit. Other potential uses include i-voting (voting over the Internet) [CRS04].

In particular, we use the BCM to construct a mechanism for transmitting alerts undetectably in a distributed system. We consider a powerful attacker who observes all message traffic and is capable of reading, replacing, and delaying circulating

Windows security updates, while a user with a broadband connection probably will not object.

messages. Occasionally, we may need to issue an alert to *all* network nodes. Our algorithms allow an alert to spread through a network at the full speed of an epidemic, while remaining undetectable to the attacker. As the alert percolates across the network, all nodes unwittingly come to possess it, making it especially difficult to identify the originator even if the secret key is compromised and the attacker can inspect the nodes' final states.

1.3 Verifiable Random Function

Random bits are an essential part of many distributed protocols. One way to generate random bits in practice is to use a **pseudo-random function** (PRF) [GGM86], whose outputs resemble random bit-strings as long as secret key SK is unknown. Unfortunately, PRFs have a limitation that the owner of the secret key must be trusted to correctly compute a given PRF value.

Our next tool, called the **verifiable random function** (VRF) [MRV99], is a pseudo-random function that also provides a proof π_x for the correctness of its output $F_{SK}(x)$ at some point x. VRFs have found numerous applications in protocol design [MR01,MR02,JS04], yet they remain not very well studied. There exist only a handful of VRF constructions in the standard model [MRV99,Lys02,Dod03], which are neither very practical nor suitable for use in protocols. Existing VRF (and PRF) constructions process their inputs bit-by-bit and may require multiple arithmetic operations. What's worse, the size of their proofs π_x and keys (PK, SK) grows very large as the input length increases.

We construct a simple and efficient VRF, which has constant size proofs and keys for all input sizes. Trivially, we also obtain the first PRF in the standard model, which does not process its inputs bit-by-bit. Our construction can be instantiated with elliptic curve groups of reasonable size, which makes it quite practical. Our PRF and VRF admit efficient zero-knowledge proofs for statements of the form "$y = F_{SK}(x)$" and "$y \neq F_{SK}(x)$" given a commitment to the secret key SK. These desirable characteristics prompted Camenisch *et al.* [CHL05] to use our VRF in their offline anonymous e-cash scheme and Chow *et al.* [CHYC05] to use it in their distributed electronic lottery.

1.4 Threshold Pseudo-Random Permutations

Traditionally, a **pseudo-random permutation** (PRP) has been used to model secure block ciphers like the DES (U.S. Data Encryption Standard). Its outputs look random to parties without the secret key. However, PRP's (unlike PRF's) outputs always possess unique inverses. In their celebrated paper, Luby and Rackoff [LR88] showed that a $2l$-bit PRP can always be constructed from four l-bit PRFs.

The security of a PRP (and of other cryptographic tools) relies on the owner of the secret key, who has a primary responsibility of keeping the key safe. Alas, it is not always realistic to entrust the secret key to a single party. The last tool is a **threshold pseudo-random permutation**, which allows parties holding shares of the secret key SK to jointly encrypt and decrypt circulating messages. Many protocols are defined for PRPs and, when needed, can now be readily translated into the distributed setting. This makes sense for sensitive operations like key-encrypting-key in the Key Distribution Center. Applications such as distributed remotely keyed authenticated encryption and CBC (cipherblock chaining) mode become possible, since they require a PRP as a building block (regular PRFs do not suffice).

Our main technique for constructing threshold PRPs is a distributed Luby-Rackoff construction. Our protocol uses only $O(1)$ communication rounds and invokes the multiplication protocol for the underlying secret sharing scheme $O(mn + m \log m)$ times, where n is the number of servers and m is the maximum input length). The protocol tolerates up to $\tau = \lfloor (n-1)/2 \rfloor$ dishonest servers in the semi-honest model, which is consistent with some prior work on distributed PRFs [NPR99] and multiparty tools [ACS02, CGH00, DFK$^+$06, Kil05] used in our constructions. It can be made robust using standard techniques [RBO89] and, as we show, can be amended to ensure **proactive security** [HJKY95].

Luby-Rackoff intermediate computations must be kept private from dishonest servers during the protocol, yet we must still evaluate the round PRFs on them. As a result, we share both the secret keys and the intermediate inputs among the n servers. We give a distributed protocol for sharing the computation of our PRF and VRF (from previous section). Effectively, we implement **oblivious distributed PRFs**, where servers do not learn what the input is, yet blindly compute the PRF value.

1.5 Economic Model of Network Security

As we have explained, the overall security of a distributed system is very much dependent on the behavior of individual servers. Traditionally, servers are classified as either honest or malicious. We ask whether the system is adversely affected when servers act selfishly.

In our model, there are n servers, each of which may or may not be protected against viruses, that are connected by a network in the form of a graph, and any virus that infects some server eventually infects all of its unprotected neighbors. If anti-virus software is available, a natural response would be to protect all the servers—but perhaps the anti-virus software itself creates costs, both in money and time to purchase and install the software and in reduced efficiency or usability of the protected server. Suppose that protecting a server by installing anti-virus software costs the owner of the server C, but that having the server be infected costs L, which may or may not be greater than C. If the virus spreads from some initial server chosen uniformly at random, on which servers does it make sense to install the anti-virus software?

The answer will depend on whether we adopt the perspective of the owner of a single server or of the society as a whole. When the anti-virus software costs more than the loss from infection, no economically rational server's owner will install the anti-virus software, every server will be infected, and the system will incur a social cost of Ln. But for many graphs, selective inoculation of a few central servers can limit the spread of infection to a small subset of the graph, greatly reducing the total cost of infection in return for a small investment in anti-virus software.

We prove many game-theoretic properties of our model, including an easily applied characterization of Nash equilibria[2], culminating in our showing that a centralized solution can give a much better total cost than an equilibrium solution. Though it is **NP**-hard to compute such a social optimum, we show that the problem can be reduced to a previously unconsidered combinatorial problem that we call the **sum-of-squares partition problem**. Using a greedy algorithm based on sparse cuts, we show that this problem can be approximated to within a factor of $O(\log^2 n)$.

1.6 Organization of Dissertation

The rest of the dissertation is structured as follows. Chapter 2 covers technical preliminaries necessary to understand this thesis. The following chapters are pre-

[2]Roughly speaking, in a non-cooperative game, a Nash equilibrium is a collective strategy, where no player can benefit by changing only his own strategy.

sented in a self-contained manner and can be read independently. In Chapter 3, we construct the blind coupon mechanism and describe its applications to distributed systems. In Chapter 4, we give a verifiable random function with constant-size keys and proofs. We use this verifiable random function to construct the first reasonably efficient threshold pseudo-random permutation in Chapter 5. Chapter 6 focuses on incentive compatibility in the context of distributed systems. Finally, in Chapter 7, we summarize our results and describe some open problems.

Chapter 2

Technical Preliminaries

This chapter introduces the notation and reviews some basic definitions. The aim is to make the dissertation self-contained.

Let k be a security parameter. As customary, we model the protocol participants by probabilistic Turing machines whose running time is polynomial in k (abbreviated as PPTs).

2.1 Notation

We use standard notation from [BSMP91].

Let $A(\cdot)$ be an algorithm. Then $y \leftarrow A(x)$ denotes that y was obtained by running A on input x. In case A is deterministic, this y is unique. If A is probabilistic, then y is a random variable. If S is a finite set, then $y \overset{\$}{\leftarrow} S$ denotes that y was chosen from S uniformly at random.

By $A^{\mathcal{O}}(\cdot)$, we denote an algorithm that has oracle access to $\mathcal{O}$. It can be visualized as a Turing machine with a black box that can decide certain decision problems in a single step. The oracle will compute a function that we would not be able to compute on our own.

Let b be a Boolean predicate. The notation $[b(y) \mid y \leftarrow A(x)]$ denotes the event that $b(y)$ is true after y is output by A on input x. Similarly, the statement

$$\Pr\left[b(x_0,\ldots,x_n) \mid x_0 \overset{\$}{\leftarrow} S, x_1 \leftarrow A_1(x_0),\ldots,x_n \leftarrow A_n(x_0,\ldots,x_{n-1})\right]$$

denotes the probability that the predicate $b(x_1,\ldots,x_n)$ will be true after the sequential execution of algorithms A_i. Here, x_0 is drawn uniformly from S and subsequent x_i are drawn from distributions of algorithms A_i, possibly depending on previous inputs.

Finally, we will say that $negl(k) : \mathbb{N} \mapsto (0,1)$ is a **negligible function** if for every $c > 0$, for all sufficiently large k, $negl(k) < 1/k^c$

2.2 Groups

We define some groups that are used in our constructions throughout the thesis. In Section 2.2.2, we review properties of elliptic curve groups. Then in Section 2.2.3, we define a class of bilinear groups.

Recall that a group $(\mathbb{G}, \circ)$ is a nonempty set $\mathbb{G}$ together with a binary operation $\circ : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}$ satisfying the group axioms: (a) $\forall a, b, c \in \mathbb{G}, (a \circ b) \circ c = a \circ (b \circ c)$ (**associativity**); (b) $\exists e \in \mathbb{G}, \forall a \in \mathbb{G}, e \circ a = a \circ e = a$ (**identity element**); (c) $\forall a \in \mathbb{G}, \exists b \in \mathbb{G}, a \circ b = b \circ a = e$ (**inverse element**).

2.2.1 Groups $\mathbb{Z}_n$ and $\mathbb{Z}_n^*$

The set of integers modulo n, denoted $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$, forms a group under addition modulo n. Another important group is $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid \gcd(x, n) = 1\}$, the set of integers relatively prime to n, together with multiplication modulo n.

We denote additive shares over $\mathbb{Z}_n$ of a value $a \in \mathbb{Z}_n$ by $\langle a \rangle_1^P, \ldots, \langle a \rangle_k^P \in \mathbb{Z}_n$; *i.e.* $a = \sum_{j=1}^{k} \langle a \rangle_j^P \bmod n$. Meanwhile, we denote Shamir shares [Sha79] of $a \in \mathbb{Z}_n$ by $[a]_1^P, \ldots, [a]_k^P \in \mathbb{Z}_n$; *i.e.* $a = \sum_{j=1}^{\tau} \lambda_j [a]_j^P \bmod n$, where τ is the threshold and λ_j are the Lagrange coefficients. We will denote the set of quadratic residues modulo n by $QR(n) = \{x^2 \bmod n \,|\, x \in \mathbb{Z}_n^*\}$. It is well-known that it forms a group too.

2.2.2 Elliptic Curve Groups

Let n be an integer greater than 1 and not divisible by 2 or 3. We first introduce projective coordinates over $\mathbb{Z}_n$. Consider the set $\bar{U}$ of triples $(x, y, z) \in \mathbb{Z}_n^3$ satisfying $\gcd(x, y, z, n) = 1$. Let $\sim$ be the equivalence relation on $\bar{U}$ defined by $(x, y, z) \sim (x', y', z')$ iff there exists $\lambda \in \mathbb{Z}_n^*$ such that $(x, y, z) = (\lambda x', \lambda y', \lambda z')$. Let U be the set of equivalence classes in $\bar{U}$. We denote the equivalence class of (x, y, z) as $(x : y : z)$.

An elliptic curve over $\mathbb{Z}_n$ is defined by the equation

$$E : Y^2 Z \equiv X^3 + aXZ^2 + bZ^3 \pmod{n},$$

where a, b are integers satisfying $\gcd(4a^2 - 27b^3, n) = 1$. The set of points on $E/\mathbb{Z}_n$ is the set of equivalence classes $(x : y : z) \in U$ satisfying $y^2 z \equiv x^3 + axz^2 + bz^3 \pmod{n}$, and is denoted by $E(\mathbb{Z}_n)$. Note that if n is prime, these definitions correspond to the usual definitions for projective coordinates over prime fields. An elliptic curve can be made into a group by defining a suitable operation on its points. We introduce a "point at infinity" $\mathcal{O}$, which will be the identity element of the group. To add two points P_1 and P_2, we draw a straight line through them and compute the third point of intersection P_3. Then,

$$P_1 + P_2 + P_3 = \mathcal{O}$$

gives the point at infinity, and so we let $P_1 + P_2 = -P_3$ (see Figure 2.1).

Let p and q be primes, and let $n = pq$. Let $E_p : Y^2Z = X^3 + a_pXZ^2 + b_pZ^3$ and $E_q : Y^2Z = X^3 + a_qXZ^2 + b_qZ^3$ be elliptic curves defined over $\mathbb{F}_p$ and $\mathbb{F}_q$, respectively. We can use the Chinese remainder theorem to find a and b yielding an elliptic curve $E : Y^2Z = X^3 + aXZ^2 + bZ^3$ over $\mathbb{Z}_n$ such that the reduction of E modulo p gives E_p and likewise for q.

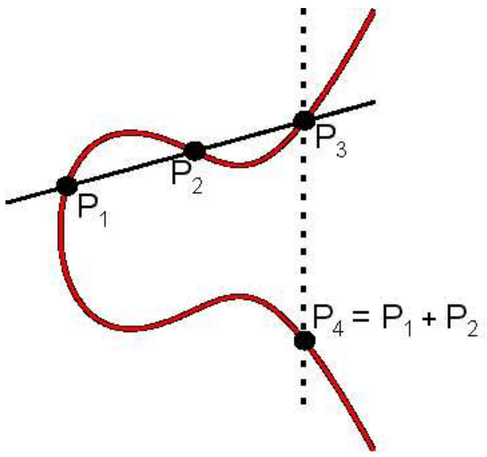

Figure 2.1: Elliptic curve addition

It can also be shown that the Chinese remainder theorem gives a set isomorphism

$$E(\mathbb{Z}_n) \xrightarrow{\sim} E_p(\mathbb{F}_p) \times E_q(\mathbb{F}_q)$$

inducing a group operation on $E(\mathbb{Z}_n)$. For almost all points in $E(\mathbb{Z}_n)$, the usual group operation formulae for the finite field case will compute the induced group operation. When they fail, the attempted operation gives a factorization of the composite modulus n. Unless $E_p(\mathbb{F}_p)$ or $E_q(\mathbb{F}_q)$ has smooth or easily guessable order, this will happen only with negligible probability (see [Gal02] for more details).

2.2.3 Bilinear Groups

Many of our constructions operate on groups equipped with bilinear maps. We briefly review their properties below.

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two (multiplicative) cyclic groups of prime order p. Also, let g be a generator of $\mathbb{G}$. We say that a group $\mathbb{G}$ is bilinear if the group action in $\mathbb{G}$ is efficiently computable and there exists an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$.

We shall call a map **bilinear** if it is linear with respect to each of its variables. Formally:

Definition 1 *An (admissible) bilinear map* $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$ *is a map with the following properties:*

1. **Bilinear:** *for all* $u, v \in \mathbb{G}$ *and* $x, y \in \mathbb{Z}$, *we have* $e(u^x, v^y) = e(u, v)^{xy}$.
2. **Non-degenerate:** $e(g, g) \neq 1$.
3. **Computable:** *there is an efficient algorithm to compute* $e(u, v)$ *for all* $u, v \in \mathbb{G}$.

Such maps can be constructed from Weil and Tate pairings on elliptic curves or Abelian varieties [BF01a, JN01, Gal01].

2.3 Some Hard Problems

The basic ingredient in most cryptographic tools, including the ones we propose, is a difficult computational problem. Intuitively, the "problem $\mathcal{P}$ is hard" if no algorithm can find its solution using a reasonable amount of resources (measured in time, space, etc.).

In this section, we give a high-level overview of computationally hard problems encountered in this thesis. Our proofs of security rely on well-established problems such as factoring, discrete logarithm, and Diffie-Hellman problems (Section 2.3.1). In addition, we also introduce several new problems, which appear to be hard (Section 2.3.2).

2.3.1 Standard Assumptions

Problem	Abbreviation	Input	Output
Factoring	FACTOR	$n = pq$	p, q
Discrete Logarithm	DL	(g, g^x)	x
Computational Diffie-Hellman	CDH	(g, g^x, g^y)	g^{xy}
Decisional Diffie-Hellman	DDH	(g, g^x, g^y, R)	is $R = g^{xy}$?

Table 2.1: Well-known computationally hard problems.

Table 2.1 summarizes some standard problems, which resisted efficient solutions for the last two decades. We describe them in more detail below.

The **factorization problem** asks: given integer $n > 0$ to find distinct primes p_i together with exponents $e_i \geq 0$ such that $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. While a computer can easily calculate the product of large numbers, factoring large numbers is much more difficult. The most obvious algorithm for factoring n is trial division by primes smaller than $\sqrt{n}$. This computation is only feasible for small numbers that are much smaller than the ones used in cryptography. More efficient algorithms exist such as Pollard's ρ [Pol75] and quadratic sieve [BLZ94] algorithms, but they all require subexponential running time and would take months to factor 1024-bit long numbers used in cryptography.

Suppose $\mathbb{G}$ is a cyclic group, whose generator is $g \in \mathbb{G}$. Another important problem is the **discrete logarithm problem** (DL) on $\mathbb{G}$. It asks: given g and element $h \in \mathbb{G}$, find the unique number x ($0 \leq x \leq |\mathbb{G}| - 1$) such that $h = g^x$. This number is denoted by $x = \log_g h$. The problem is easy for integers, but when we work in a slightly different setting, it becomes very hard. As of today, we do not know how to efficiently compute discrete logarithms in the quadratic residues subgroup of $\mathbb{Z}_p^*$ ($p = 2q + 1$) and in many elliptic curve groups. The naïve algorithm that computes successive powers $g^0, g^1, g^2, \ldots$ until element h is obtained will take too long for large groups. More efficient algorithms exist (most notably, the index calculus algorithm [COS86]), but they run in subexponential time, which is still not very efficient.

The **computational Diffie-Hellman problem** (CDH) is closely related to the DL problem. It was introduced in a seminal paper by Diffie and Hellman [DH76], who used it to construct a key exchange protocol over an insecure channel. The CDH problem is the following: given (g, g^x, g^y) (but not x nor y), compute g^{xy}. This problem is considered hard, and in some instances it is as hard as the discrete logarithm problem [Mau94]. In fact, the best known method today for breaking the CDH problem is computing discrete logarithm of g^x or g^y.

An interesting fact about CDH is that there is no efficient algorithm to recognize a solution to the problem. This brings up the **decisional Diffie-Hellman problem**(DDH), which is the following: given (g, g^x, g^y) and random $R \in \mathbb{G}$, decide whether $R = g^{xy}$ or not. The DDH problem is very attractive and has been used to construct many cryptographic primitive such as the widely used ElGamal encryption scheme [ElG85].

2.3.2 New Assumptions

Our tools also rely on several Diffie-Hellman-type assumptions, which are less studied. These assumptions are summarized in Table 2.2.

As before, we let $\mathbb{G} = \langle g \rangle$ be a cyclic group. We consider the **q-Diffie-Hellman inversion problem** (q-DHI), which asks: given the values $\left(g, g^x, \ldots, g^{(x^q)}\right)$ for a random x ($0 \leq x \leq |\mathbb{G}| - 1$), compute $g^{1/x}$.

The decisional analogue is the **q-decisional Diffie-Hellman inversion problem** (q-DDHI), which asks given the same input, to distinguish $R = g^{1/x}$ from a random element in $\mathbb{G}$.

Some of our tools operate on bilinear groups $\mathbb{G}$ with bilinear maps $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$. Note that bilinear maps provide an algorithm for solving the DDH problem in $\mathbb{G}$: Given (g, g^x, g^y, R), we can determine if $R = g^{xy}$ by checking that $e(g^x, g^y) = e(g, R)$. We can also break the q-DDHI problem on bilinear groups by checking if $e(g, g) = e(g^x, R)$. Since the DDH and q-DDHI problems are easy, we cannot use them to build cryptographic tools in bilinear groups. Instead, we can put to use in security the hardness of appropriate computational assumptions.

We also use a stronger version of the q-DDHI problem, called the **q-decisional bilinear Diffie-Hellman inversion problem** (q-DBDHI), which holds in bilinear groups. Given the same input $\left(g, g^x, \ldots, g^{(x^q)}\right)$ for a random x, the q-DDHI asks to distinguish $R = e(g, g)^{1/x}$ from a random element in $\mathbb{G}_1$.

Problem	Abbreviation	Input	Output
DH Inversion	q-DHI	$\left(g, g^x, \ldots, g^{(x^q)}\right)$	$g^{1/x}$
DDH Inversion	q-DDHI	$\left(g, g^x, \ldots, g^{(x^q)}, R\right)$	is $R = g^{1/x}$?
Bilinear DDH Inversion	q-DBDHI	$\left(g, g^x, \ldots, g^{(x^q)}, R\right)$	is $R = e(g, g)^{1/x}$?
Subgroup Membership		$(\mathbb{G}, \mathbb{D}, g, d, R)$	is $R \in \mathbb{D}$?
Subgroup Escape		$(\mathbb{G}, \mathbb{D}, d)$	$R \in \mathbb{G} \setminus \mathbb{D}$

Table 2.2: Some new computationally hard problems.

In this thesis, we also consider a class of general mathematical problems, which deal with subgroups. If $\mathbb{G}$ is a cyclic group and $\mathbb{D}$ is its subgroup, then determining the membership of an element of $\mathbb{G}$ in its subgroup is not always easy.

The **subgroup membership problem** asks given generators for $\mathbb{G}$ and $\mathbb{D}$, to decide if a given element $y \in \mathbb{G}$ belongs to $\mathbb{D}$. Many problems can be categorized as subgroup membership problems [CS02, GM82, NS98, OU98, Pai99, NBD01]. For example, the DDH problem asks on input (g, g^x, g^y, R) to decide if $R = g^{xy}$. This is equivalent to asking if (g^x, R) is in the subgroup of $\mathbb{G} \times \mathbb{G}$ generated by (g, g^y).

The **subgroup escape problem** asks: given a generator for $\mathbb{D}$ (but not $\mathbb{G}$), find an element of $\mathbb{G} \setminus \mathbb{D}$. These problems appear hard in elliptic groups over composite modulus $n = pq$.

2.4 Generic Group Model

Whenever we make an assumption that problem $\mathcal{P}$ appears hard, we back up our intuition by establishing lower bounds on solving $\mathcal{P}$ in the generic group model. This model was introduced by Shoup [Sho97] for measuring the exact difficulty of solving discrete logarithm problems. Algorithms in generic groups do not exploit any properties of the encodings of group elements. They can access group elements only via a random encoding algorithm that encodes group elements as random bit-strings.

Formally, let $\mathbb{G}$ be a finite cyclic group and let $U \subseteq \{0,1\}^*$ be a set such that $|U| \geq |\mathbb{G}|$. We define a random injective function $\sigma : \mathbb{G} \to U$, which maps group elements to their string representations. Generic algorithms have access to an oracle computing group action. On input of $x \pm y$, the oracle returns $\sigma(\sigma^{-1}(x) \pm \sigma^{-1}(y))$ when both $x, y \in \sigma(\mathbb{G}) \subseteq U$, and otherwise the special symbol $\perp$. Algorithms can use the oracle to decide whether $x \in U$ is in $\sigma(\mathbb{G})$ or not by sending the query $x + x$ to the oracle. If $x \notin \sigma(\mathbb{G})$, the reply will be $\perp$. When $\mathbb{G}$ is a bilinear group with a mapping $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$, we provide two encoding functions, $\sigma : \mathbb{G} \mapsto U$ and $\sigma_1 : \mathbb{G}_1 \mapsto U$, where $|U| \geq |\mathbb{G}| + |\mathbb{G}_1|$. Generic algorithms will have access to oracles computing group actions in $\mathbb{G}, \mathbb{G}_1$ and to an extra oracle that on input of (x, y) returns $\sigma_1(e(\sigma^{-1}(x), \sigma^{-1}(y)))$ when $x, y \in \sigma(\mathbb{G})$.

Proofs in the generic group model provide heuristic evidence of problem $\mathcal{P}$'s hardness when an attacker does not take advantage of group elements' encoding. However, they do not necessarily say anything about the difficulty of problem $\mathcal{P}$ in a concrete group. For example, Shoup showed that DL problem in $\mathbb{Z}_n$ cannot be solved by a generic algorithm in $o(\sqrt{n})$ time. Yet, the well-known index calculus algorithm [COS86] can compute discrete logarithms in $\mathbb{Z}_n$ faster than a generic algorithm can.

In this thesis, $\mathbb{G}$ will usually be a bilinear or an elliptic curve group with a very complicated encoding of its elements. We thus expect lower bounds on generic security of our assumptions to be comparable to their lower bounds in this group.

2.5 Secure Multi-Party Computation

Some of our cryptographic tools fall under a fairly general umbrella of **secure multi-party computation** (MPC) [Gol98]. In this section, we provide a brief overview of the MPC problem. We also explain why generic MPC solutions can hardly be considered as practical alternatives for our results.

The MPC problem involves n servers $P_1, \ldots, P_n$, where each server P_i has a private input x_i. The servers want to evaluate a function $f(x_1, \ldots, x_n)$ on their inputs. Two constraints must be satisfied: (i) servers should learn only the output and nothing else (**privacy**) and (ii) each server should receive a correct output (**correctness**). Ideally, the servers could just hand their inputs to a trusted party who would evaluate the function on their behalf. Of course in the real world, no such trusted party exists. Secure multi-party computation allows to simulate the existence of such trusted party despite interference from corrupt servers.

Completeness theorems by Yao [Yao82] and by Goldreich, Micali, and Wigderson [GMW87] showed that *any* probabilistic function $f(x_1, \ldots, x_n)$ can be securely (privately and correctly) computed by an honest majority of servers if trapdoor permutations exist. Later, Ben-Or, Goldwasser, and Wigderson [BoGW88] generalized these results without making use of cryptography. The basic idea is to start with an algorithm for computing $f(x_1, \ldots, x_n)$ and convert this algorithm into a corresponding Boolean circuit. Then for each gate in the circuit, the servers perform a distributed multiplication protocol, and compute Shamir shares [Sha79] of that gate's output. In this way, the servers interact working their way up the circuit from leaves to root, eventually obtaining shares of $y = f(x_1, \ldots, x_n)$.

The practicality of cryptographic tools depends heavily on the amount of time needed to complete the protocol (**bit complexity**), the amount of generated message traffic (**communication complexity**), and the number of communication rounds (**round complexity**), where in one round each server can send a message to every other server. Unfortunately, generic MPC solutions are as impractical as they are beautiful. Even though the interaction for each gate can be implemented in $O(1)$ rounds, the total number of communication rounds will still be linear in the depth of the circuit. Consequently, MPC solutions to our cryptographic primitives require a prohibitive number of communication rounds. By focusing on special important problems rather than a general class of functions, we construct efficient and non-interactive cryptographic tools.

Chapter 3

Blind Coupon Mechanism[*]

In this chapter, we introduce a new cryptographic primitive called the **blind coupon mechanism** (BCM).

3.1 Overview

The BCM is related, yet quite different, from the notion of commitment. It consists of a set D_{SK} of **dummy coupons** and a set S_{SK} of **signal coupons** ($D_{SK} \cap S_{SK} = \emptyset$). The owner of the secret key SK can efficiently sample these sets and distinguish between their elements. We call the set of dummy and signal coupons, $D_{SK} \cup S_{SK}$, the set of **valid** coupons.

The BCM comes equipped with a **verification algorithm** $\mathcal{V}_{PK}(x)$ that checks if x is indeed a valid coupon. There is also a probabilistic **combining algorithm** $\mathcal{C}_{PK}(x, y)$, that takes as input two valid coupons x, y and outputs a new coupon which is, with high probability, a signal coupon if and only if at least one of the inputs is a signal coupon. As suggested by the notation, both algorithms can be computed by

[*]This chapter describes joint work with James Aspnes, Zoë Diamadi, Kristian Gjøsteen, and René Peralta [ADG+05].

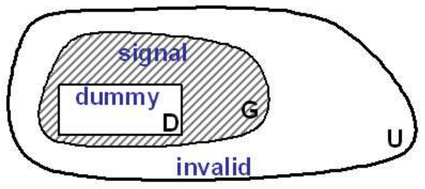

Figure 3.1: Abstract group structure used in our BCM construction.

anyone who has access to the public key PK of the blind coupon mechanism.

We regard the BCM secure if an observer who lacks the secret key SK (a) cannot distinguish between dummy and signal coupons (**indistinguishability**); (b) cannot engineer a new signal coupon unless he is given another signal coupon as input (**unforgeability**); and (c) cannot distinguish randomly chosen coupons from coupons produced by the combining algorithm (**blinding**).

Overview of Our Constructions. Our BCM construction uses an abstract group structure (U, G, D). Here, U is a finite set, $G \subseteq U$ is a cyclic group, and D is a subgroup of G. The elements of D will represent dummy coupons and the elements of $G \setminus D$ will be signal coupons (see also Figure 3.1). The combining operation will simply be a group operation. To make verification possible, there will need to be an easy way to distinguish elements of G (valid coupons) from elements of $U \setminus G$ (invalid coupons).

In order for the BCM to be secure, the following two problems must be hard on this group structure:

- **Subgroup Membership Problem**: Given generators for G and D and an element $y \in G$, decide whether $y \in D$ or $y \in G \setminus D$.
- **Subgroup Escape Problem**: Given a generator for D (but not G), find an element of $G \setminus D$.

The subgroup membership problem has appeared in many different forms in the literature [CS02, GM82, NS98, OU98, Pai99, Gjø05, NBD01]. The subgroup escape problem has not been studied before. To provide more confidence in its validity, we later analyze it in the generic group model.

Notice that the task of distinguishing a signal coupon from a dummy coupon (indistinguishability) and the task of forging a signal coupon (unforgeability) are essentially the subgroup membership and subgroup escape problems. The challenge thus becomes to find a concrete group structure (U, G, D) for which the subgroup membership and the subgroup escape problems are hard.

We provide two instantiations of the group structure: one using groups with bilinear pairings, and one using elliptic curves over composite moduli.

Spreading Alerts with the BCM. The BCM can potentially be useful in various applications. In this chapter, we discuss one such application of propagating alerts quickly and quietly throughout the network. During the initial network setup, the network administrator generates the BCM's public and secret keys. He then distributes signal coupons to **sentinel nodes**, which run Intrusion Detection Software. All other nodes receive dummy coupons. In our mechanism, nodes continuously transmit either dummy or signal coupons with all nodes initially transmitting dummy coupons. Sentinel nodes switch to sending signal coupons when they detect the attacker's presence. The BCM's combining algorithm allows dummy and signal coupons to be combined so that a node can propagate signal coupons without having to know that it has received any, and so that an attacker (who can observe all message traffic) cannot detect where or when signals are being transmitted within the stream of dummy messages.

In addition, the BCM's verification algorithm defends against Byzantine nodes [LSP82]:

While Byzantine nodes can replay old dummy messages instead of relaying signals, they cannot flood the network with invalid coupons, thereby preventing an alert from spreading; at worst, they can only act like crashed nodes.

We prove that if the underlying BCM is secure, then the attacker cannot distinguish between executions where an alert was sent and executions where no alert was sent. The time to spread the alert to all nodes will be determined by the communications model and alert propagation strategy. At any point in time, the network administrator can sample the state of some network node and check if it possesses a signal coupon.

Chapter Organization. The rest of the chapter is organized as follows. In Section 3.2, we formally define the notion of a blind coupon mechanism and sketch an abstract group structure, which will allow us to implement it. Then in Section 3.3, we provide two concrete instantiations of this group structure using certain bilinear groups and elliptic curves over the ring $\mathbb{Z}_n$. In Section 3.4, we show how the BCM can be used to spread alerts quietly throughout a network. In Section 3.5, we analyze the hardness of the subgroup escape problem in the generic group model. We conclude with a discussion of related work in Section 3.6.

3.2 Blind Coupon Mechanism

The critical component of our algorithms that allows information to propagate undetectably among the processes is a cryptographic primitive called a **blind coupon mechanism** (BCM). In Section 3.2.1, we give a formal definition of the BCM and its security properties. In Section 3.2.2, we describe an abstract group structure that will allow us to construct the BCM.

3.2.1 Definitions

Definition 2 *A **blind coupon mechanism** is a tuple of PPT algorithms* $(\mathcal{G}, \mathcal{V}, \mathcal{C}, \mathcal{D})$ *in which:*

- $\mathcal{G}(1^k)$*, the probabilistic **key generation algorithm**, outputs a pair of public and secret keys* (PK, SK) *and two strings* (d, s)*. The public key defines a universe set* U_{PK} *and a set of **valid coupons*** G_{PK}*. The secret key implicitly defines an associated set of **dummy coupons*** D_{SK} *and a set of **signal coupons*** S_{SK}*.*[1] *It is the case that* $d \in D_{SK}$ *and* $s \in S_{SK}$*,* $D_{SK} \cap S_{SK} = \emptyset$*, and* $D_{SK} \cup S_{SK} = G_{PK}$*.*

- $\mathcal{V}_{PK}(y)$*, the deterministic **verification algorithm**, takes as input a coupon* y *and returns 1 if* y *is valid and 0 if it is invalid.*

- $z \leftarrow \mathcal{C}_{PK}(x, y)$*, the probabilistic **combining algorithm**, takes as input two valid coupons* $x, y \in G_{PK}$ *and produces a new coupon* z*. The output* z *is a signal coupon (with overwhelming probability) whenever one or more of the inputs is a signal coupon, otherwise it is a dummy coupon (see Figure 3.2).*

- $\mathcal{D}_{SK}(y)$*, the deterministic **decoding algorithm**, takes as input a valid coupon* $y \in G_{PK}$*. It returns 0 if* y *is a dummy coupon and* 1 *if* y *is a signal coupon.*

The BCM may be established either by an external trusted party or jointly by the application participants, running the distributed key generation protocol (*e.g.* one could use a variant of [ACS02]). We assume a trusted dealer (the network administrator) who runs the key generation algorithm and distributes signal coupons to sentinel nodes at the start of the system execution. In a typical algorithm, the

[1] Note that membership in S_{SK} and D_{SK} should not be efficiently decidable when given only PK (unlike membership in G_{PK}). However, we require that membership is always efficiently decidable when given SK.

x	y	$\mathcal{C}(x, y)$
D_{SK}	D_{SK}	D_{SK}
D_{SK}	S_{SK}	S_{SK}
S_{SK}	D_{SK}	S_{SK}
S_{SK}	S_{SK}	S_{SK}

Figure 3.2: Properties of the combining algorithm.

nodes will continuously exchange coupons with each other. The combining algorithm $\mathcal{C}_{PK}$ enables nodes to locally and efficiently combine their coupons with coupons of other nodes. The verification function $\mathcal{V}_{PK}$ prevents the adversary from flooding the system with invalid coupons and making it impossible for the signal to spread.

For this application, we require the BCM to have certain specific security properties.

Definition 3 *We say that a blind coupon mechanism* $(\mathcal{G}, \mathcal{V}, \mathcal{C}, \mathcal{D})$ *is* ***secure*** *if it satisfies the following requirements:*

1. ***Indistinguishability****: Given a valid coupon* y*, the adversary cannot tell whether it is a signal or a dummy coupon with probability better than* $1/2$*. Formally, for any PPT algorithm* $\mathcal{A}$*,*

$$\left| \Pr\left[b = b' \;\middle|\; \begin{array}{c} (PK, SK, d, s) \leftarrow \mathcal{G}(1^k); \\ x_0 \xleftarrow{\$} D_{SK}; x_1 \xleftarrow{\$} S_{SK}; \\ b \xleftarrow{\$} \{0,1\}; b' \leftarrow \mathcal{A}(1^k, PK, d, x_b) \end{array} \right] - \frac{1}{2} \right| \leq negl(k)$$

2. ***Unforgeability****: The adversary is unlikely to fabricate a signal coupon without the use of another signal coupon as input*[2]*. Formally, for any PPT algorithm*

[2]The adversary, however, can easily generate polynomially many dummy coupons by using $\mathcal{C}_{PK}(\cdot, \cdot)$ with the initial dummy coupon d that he receives.

$\mathcal{A}$,

$$\Pr\left[\, y \in S_{SK} \;\middle|\; \begin{array}{c} (PK, SK, d, s) \leftarrow \mathcal{G}(1^k); \\ y \leftarrow \mathcal{A}\left(1^k, PK, d\right) \end{array} \right] \leq \mathit{negl}(k)$$

3. ***Blinding**: The combination $\mathcal{C}_{PK}(x, y)$ of two valid coupons x, y looks like a random valid coupon. Formally, fix some pair of keys (PK, SK) outputted by $\mathcal{G}(1^k)$. Let U_D be a uniform distribution on D_{SK} and let U_S be a uniform distribution on S_{SK}. Then, for all valid coupons $x, y \in G_{PK}$,*

$$\begin{cases} \mathrm{Dist}(\mathcal{C}_{PK}(x, y), U_D) = \mathit{negl}(k) & \text{if } x, y \in D_{SK}, \\ \mathrm{Dist}(\mathcal{C}_{PK}(x, y), U_S) = \mathit{negl}(k) & \text{otherwise.} \end{cases}$$

(Here, $\mathrm{Dist}(A, B) \overset{def}{=} \frac{1}{2}\sum_x |\Pr[A = x] - \Pr[B = x]|$ is the statistical distance between a pair of random variables A, B.)

To build the reader's intuition, we describe a straw-man construction of a BCM. Suppose we are given any semantically secure encryption scheme $\mathcal{E}(\cdot)$ and a set-homomorphic signature scheme $\mathrm{SIG}(\cdot)$ by Johnson *et al.* [JMSW02]. This signature scheme allows anyone possessing sets $x, y \subseteq \mathbb{Z}_p$ and their signatures $\mathrm{SIG}(x), \mathrm{SIG}(y)$ to compute $\mathrm{SIG}(x \cup y)$ and $\mathrm{SIG}(w)$ for any $w \subseteq x$. We represent dummy coupons by a random-length vector of encrypted zeroes; *e.g.* $x = (\mathcal{E}(0), \ldots, \mathcal{E}(0))$. The signal coupons are represented by a vector of encryptions that contains at least one encryption of a non-zero element; *e.g.* $y = (\mathcal{E}(0), \ldots, \mathcal{E}(0), \mathcal{E}(1))$. To prevent the adversary from forging coupons, the coupons are signed with the set-homomorphic signature. The combining operation is simply the set union: $\mathcal{C}_{PK}\left((x, \mathrm{SIG}(x)), (y, \mathrm{SIG}(y))\right) = \left(x \cup y, \mathrm{SIG}(x \cup y)\right)$. The drawback of this construction is immediate: as coupons are combined and passed around the network, they quickly grow very large. Constructing a BCM with no expansion of coupons is more challenging. We describe such a construction next.

3.2.2 Abstract Group Structure

We sketch the abstract group structure that will allow us to implement a secure and efficient BCM. Concrete instantiations of this group structure are provided in Section 3.3.

Let $\Gamma = \{\Gamma_k\}$ be a family of sets of tuples (U, G, D, d, s), where U is a finite set, and G is a subset of U. G also has a group structure: it is a cyclic group generated by s. D is a subgroup of G generated by d, such that the factor group G/D has prime order $|G|/|D|$. The orders of D and G/D are bounded by 2^k; moreover, $|G|/|U| \leq \mathit{negl}(k)$ and $|D|/|G| \leq \mathit{negl}(k)$.

Let $\mathcal{G}'$ be a PPT algorithm that on input of 1^k samples from Γ_k according to some distribution. We consider Γ_k to be a probability space with this distribution.

We assume there exists an efficient, deterministic algorithm for distinguishing elements of G from elements of $U \setminus G$, and an efficient algorithm for computing the group operation in G.

– The **key generation algorithm** $\mathcal{G}(1^k)$ runs $\mathcal{G}'$ to sample (U, G, D, d, s) from Γ_k, and outputs the public key $PK = (U, G, d, k)$, the secret key $SK = |D|$, as well as d and s.

  The elements of D will represent dummy coupons, the elements of $G \setminus D$ will represent signal coupons, and the elements of $U \setminus G$ will be invalid coupons (see Figure 3.1).

– The **verification algorithm** $\mathcal{V}_{PK}(y)$ checks that the coupon y is in G.

– The **combining algorithm** $\mathcal{C}_{PK}(x, y)$ is simply the group operation combined with randomization. For input $x, y \in G$, sample r_0, r_1 and r_2 uniformly at random from $\{0, 1, \ldots, 2^{2k} - 1\}$, and output $d^{r_0} \circ x^{r_1} \circ y^{r_2}$.

– Because $y^{|D|} = 1$ if and only if $y \in D$, the **decoding algorithm** $\mathcal{D}_{SK}$ checks if $y^{|D|} = 1$.

The indistinguishability and unforgeability properties of the BCM will depend on the hardness assumptions described below.

Definition 4 *The* ***subgroup membership problem*** *for* Γ *asks: given a tuple* (U, G, D, d, s) *from* Γ *and* $y \in G$*, decide whether* $y \in D$ *or* $y \in G \setminus D$*.*

The subgroup membership problem is hard if for any PPT algorithm $\mathcal{A}$,

$$\left| \Pr \left[b' = b \;\middle|\; \begin{array}{c} (U, G, D, d, s) \xleftarrow{\$} \Gamma_k; \\ y_0 \xleftarrow{\$} D; y_1 \xleftarrow{\$} G \setminus D; \\ b \xleftarrow{\$} \{0, 1\}; b' \leftarrow \mathcal{A}(U, G, D, d, s, y_b) \end{array} \right] - \frac{1}{2} \right| \leq negl(k).^3$$

Various subgroup membership problems have been extensively studied in the literature, and examples include the Decision Diffie-Hellman problem [CS02], the quadratic residue problem [GM82], among others [NS98, OU98, Pai99]. Our constructions however are more related to the problems described in [Gjø05, NBD01].

Definition 5 *The* ***subgroup escape problem*** *for* Γ *asks: given* U*,* G*,* D *and the generator* d *for* D *from the tuple* (U, G, D, d, s) *from* Γ*, find an element* $y \in G \setminus D$*.*

The subgroup escape problem is hard if for any PPT algorithm $\mathcal{A}$,

$$\Pr \left[y \in G \setminus D \;\middle|\; \begin{array}{c} (U, G, D, d, s) \xleftarrow{\$} \Gamma_k; \\ y \leftarrow \mathcal{A}(U, G, D, d) \end{array} \right] \leq negl(k).$$

[3]Henceforth, we assume that groups we operate on have some concise description, which can be passed as an argument to our algorithms. We also assume that group elements can be uniquely encoded as bit strings.

The subgroup escape problem has to our knowledge not appeared in the literature before. It is clear that unless $|G|/|U|$ is negligible, finding elements of $G \setminus D$ cannot be hard. We show in Section 3.5 that if $|G|/|U|$ is negligible, the subgroup escape problem is provably hard in the generic model.

We also note that the problem of generating a signal coupon from polynomially many dummy coupons is essentially the subgroup escape problem.

Theorem 6 *Let Γ be as above. If the subgroup membership problem and the subgroup escape problem for Γ are hard, then the corresponding BCM is secure.*

Proof: Fix k and (U, G, D, d, s) sampled from Γ_k.

We prove the blinding property first, and start with the ideal case: For input $x, y \in G$, sample r_0 uniformly from $\{0, 1, \ldots, |D| - 1\}$, and r_1 and r_2 uniformly from $\{0, 1, \ldots, |G/D| - 1\}$, and output $d^{r_0} \circ x^{r_1} \circ y^{r_2}$.

If $x, y \in D$, the product is uniformly distributed in D, since d^{r_0} is.

If $x \notin D$, then the residue class $x^{r_1}D$ is uniformly distributed in G/D. Since d^{r_0} is uniformly distributed in D, the product is uniformly distributed in G. The uniform distribution on G is $|D|/|G|$-close to the uniform distribution on $G \setminus D$. The same argument holds for y^{r_2}.

Finally we note that we do not need to know $|D|$ or $|G/D|$. Since we know that $|D|$ and $|G/D|$ are less than 2^k, sampling r_0, r_1, r_2 uniformly from the set $\{0, \ldots, 2^{2k} - 1\}$ will produce an output distribution that is 2^{-k}-close to ideal, which proves the bound for blinding

Next, we prove the indistinguishability property, so let $\mathcal{A}$ be an adversary against indistinguishability. We have a subgroup membership problem instance (U, G, D, d, s) and $y \in G$. We construct the public key $PK = (U, G, d, k)$, and give $\mathcal{A}$ as input PK, d and y.

If $\mathcal{A}$ answers 1, we conclude that $y \in G \setminus D$, otherwise $y \in D$. Whenever $\mathcal{A}$ is correct, we will be correct, so $\mathcal{A}$ must have negligible advantage.

Finally, we deal with forging. Let $\mathcal{A}$ be an adversary against unforgeability. We have a subgroup escape problem instance U, G and D, and a generator d for D. Again we construct the public key $PK = (U, G, d, k)$, and give $\mathcal{A}$ as input PK and d.

Our output is simply $\mathcal{A}$'s output. Whenever $\mathcal{A}$ succeeds, we will succeed, so $\mathcal{A}$ must have negligible success probability. ■

3.3 Constructing the BCM

We now give two instantiations of the abstract group structure (U, G, D) described in the previous section. In Section 3.3.1, we describe our BCM construction on elliptic curves over composite moduli. In Section 3.3.2, we describe an alternative BCM construction on bilinear groups. These constructions can be used to undetectably transmit a one-shot alert throughout the network. In Section 3.3.3, we describe how the BCM's bandwidth can be further expanded.

3.3.1 BCM on Elliptic Curves Modulo Composites

Let $p, q, \ell_1, \ell_2, \ell_3$ be primes, and suppose we have elliptic curves $E_p/\mathbb{F}_p$ and $E_q/\mathbb{F}_q$ such that $\#E_p(\mathbb{F}_p) = \ell_1\ell_2$ and $\#E_q(\mathbb{F}_q) = \ell_3$. Curves of this form can be found using complex multiplication techniques [BSS99, LZ94].

With $n = pq$, we can find $E/\mathbb{Z}_n$ such that $\#E(\mathbb{Z}_n) = \ell_1\ell_2\ell_3$. Let U be the projective plane modulo n, let G be $E(\mathbb{Z}_n)$, and let D be the subgroup of order $\ell_1\ell_3$. The public key is $PK = (G, D, n)$, while the secret key is $SK = (p, q, l_1, l_2, l_3)$.[4]

[4]To describe groups G and D, we publish the elliptic curve equation and the generator for D.

Verification Function For any equivalence class $(x : y : z)$ in U, it is easy to decide if $(x : y : z)$ is in $E(\mathbb{Z}_n)$ or not, simply by checking if $y^2 z \equiv x^3 + axz^2 + bz^3 \pmod{n}$.

Subgroup Membership Problem For the curve $E_p(\mathbb{F}_p)$, distinguishing the elements of prime order from the elements of composite order seems to be hard, unless it is possible to factor the group order [Gjø05].

Counting the number of points on an elliptic curve defined over a composite number is equivalent to factoring the number [HWL87, KK98]. Therefore, the group order $E_p(\mathbb{F}_p)$ is hidden.

When the group order is hidden, it cannot be factored. It therefore seems reasonable that the subgroup of $E(\mathbb{Z}_n)$ of order $\ell_1 \ell_3$ is hard to distinguish from the rest of the points on the curve, as long as the integer n is hard to factor.

Subgroup Escape Problem Anyone capable of finding a random point on the curve will with overwhelming probability be able to find a point outside the subgroup D.

Finding a random point on an elliptic curve over a field is easy: Choose a random x-coordinate and solve the resulting quadratic equation. It has rational solutions with probability close to 1/2.

This does not work for elliptic curves over the ring $\mathbb{Z}_n$, since solving square roots modulo n is equivalent to factoring n. One could instead try to choose a y-coordinate and solve for the x-coordinate, but solving cubic equations in $\mathbb{Z}_n$ seems no easier than finding square roots.

One could try to find x and y simultaneously, but there does not seem to be any

This gives away enough information to perform group operations in G, check membership in G, and generate new elements in D (but not in G).

obvious strategy. This is in contrast to quadratic curves, where Pollard [SP87] gave an algorithm to find solutions of a quadratic equation modulo a composite (which broke the Ong-Schnorr-Shamir signature system [OSS84]). These techniques do not seem to apply to the elliptic curve case.

Finding a lift of the curve over the integers does not seem promising. While torsion points are fairly easy to find, they will not exist if the curve $E/\mathbb{Z}_n$ does not have points of order less than or equal to 12. If we allow $E/\mathbb{Z}_n$ to have points of small order that are easily found, we can simply include them in the subgroup D.

Finding rational non-torsion points on curves defined over $\mathbb{Q}$ is certainly non-trivial, and seems impossibly hard unless the point on the lifted curve has small height [Sil99]. There does not seem to be any obvious way to find a lift with rational points of small height (even though they certainly exist).

What if we already know a set of points on the curve? If we are given $P_1, P_2, P_3 \in E(\mathbb{Z}_n)$, we can find, unless the points are collinear, a quadratic curve

$$C : YZ = \alpha X^2 + \beta XZ + \gamma Z^2$$

defined over $\mathbb{Z}_n$ that passes through P_1, P_2, P_3. We can find the fourth intersection point P_4 by deriving a fourth-degree polynomial in X for which we know three zeros.

To show that we could easily derive this point using the group operation, we consider the situation over the finite fields, where E and C have at most six points of intersection. Both intersect $(0 : 1 : 0)$, and since the line $Z = 0$ is a tangent to both curves in $(0 : 1 : 0)$, their intersection number in $(0 : 1 : 0)$ is greater than 1. This means that E and C intersect in exactly five points, P_1, P_2, P_3, P_4 and $(0 : 1 : 0)$.

The divisor of C is $(P_1) + (P_2) + (P_3) + (P_4) + 2((0 : 1 : 0))$. Let $C' : Z^2 = 0$ with divisor $6((0 : 1 : 0))$. Since the divisor of the function $f(X, Y, Z) = (YZ -$

$\alpha X^2 - \beta XZ - \gamma Z^2)/(Z^2)$ satisfies $\mathrm{div}(f) = \mathrm{div}(C) - \mathrm{div}(C') = 0$, we see that $(P_1) + (P_2) + (P_3) + (P_4) - 4((0:1:0)) = 0$, which means that

$$P_1 + P_2 + P_3 + P_4 = (0:1:0)$$

The fourth point is therefore the inverse sum of the three known points.

If points of the curve only yield new points via the group operation, and it seems hard to otherwise find points on $E(\mathbb{Z}_n)$, it is reasonable to assume that $E(\mathbb{Z}_n)$ and its subgroup, as described in the previous section, yield a hard subgroup escape problem.

3.3.2 BCM on Groups With Bilinear Pairings

Let p, ℓ_1, ℓ_2, and ℓ_3 be primes such that $p+1 = 6\ell_1\ell_2\ell_3$, and $p = 2 \pmod 3$. Here, l_1, l_2, l_3 must be distinct and larger than 3. The elliptic curve $E : Y^2 = X^3 + 1$ defined over $\mathbb{F}_p$ is supersingular and has order $p+1$. Because $\mathbb{F}_{p^2}^*$ has order $p^2 - 1 = (p+1)(p-1)$, there is a modified Weil pairing $e : E(\mathbb{F}_p) \times E(\mathbb{F}_p) \to \mathbb{F}_{p^2}^*$. This pairing is known to be bilinear: $e(P^a, Q^b) = e(P,Q)^{ab}$ for all $P, Q \in E(\mathbb{F}_p)$ and $a, b \in \mathbb{Z}_p$. It can be computed as described in [BF01a].

Let $U = E(\mathbb{F}_p)$, and let G and D be the subgroups of $E(\mathbb{F}_p)$ of order $\ell_1\ell_2$ and ℓ_1, respectively. We also let P be a point in $E(\mathbb{F}_p)$ of order $6\ell_1\ell_2\ell_3$, and let R be a point of order $6\ell_3$ in $E(\mathbb{F}_p)$, say $R = P^{\ell_1\ell_2}$ The public key is $PK = (G, D, p, R)$ and the secret key is $SK = (l_1, l_2, l_3)$. The pairing e allow us to describe G in the public key without giving away secret information.

Verification Function We claim that for any point $Q \in E(\mathbb{F}_p)$, $Q \in G$ if and only if $e(Q, R)$ is equal to 1. If $Q \in G$, then Q has order $\ell_1\ell_2$ and for some integer

s, $Q = P^{6s\ell_3}$. Then

$$e(Q, R) = e(P^{6s\ell_3}, P^{\ell_1\ell_2}) = e(P, P)^{6s\ell_1\ell_2\ell_3} = 1.$$

So the point R and the pairing e allows us to determine if points are in G or in $U \setminus G$.

Subgroup Membership Problem Distinguishing the subgroup D (the points of order ℓ_1) from G (the points of order $\ell_1\ell_2$) can easily be done if the integer $\ell_1\ell_2\ell_3$ can be factored. In general, factoring seems to be the best way to distinguish the various subgroups of $E(\mathbb{F}_p)$.

Because we do not reveal any points of order ℓ_2 or $\ell_2\ell_3$, it seems impossible to use the pairing to distinguish the subgroup D in this way. (Theorem 1 of [Gjø05] assumes free sampling of any subgroup, which is why it and the pairing cannot be used to distinguish the subgroups of $E(\mathbb{F}_p)$.) It therefore seems reasonable to assume that the subgroup membership problem for G and D is hard, which will provide indistinguishability.

Subgroup Escape Problem For a general cyclic group of order $\ell_1\ell_2\ell_3$, it is easy to find elements of order $\ell_1\ell_2$ if ℓ_3 is known. Unless ℓ_3 is known, it is hard to find elements of order $\ell_1\ell_2$, and knowing elements of order ℓ_1 does not help.

For our concrete situation, factoring the integer $\ell_1\ell_2\ell_3$ into primes seems to be the best method for solving the problem. If the primes ℓ_1, ℓ_2 and ℓ_3 are chosen carefully to make the product $\ell_1\ell_2\ell_3$ hard to factor, it seems reasonable to assume that the subgroup escape problem for U, G and D is hard.

3.3.3 Extending the BCM's Bandwidth

The blind coupon mechanism allows to undetectably transmit a single bit. Although this is sufficient for our network alert application, sometimes we may want to transmit longer messages.

Trivial Construction. By using multiple blind coupon schemes over different moduli in parallel, we can transmit longer messages. Each m-bit message $x = x_1 \dots x_m$ is represented by a vector of coupons $\langle c_1, \dots, c_{2m} \rangle$, where each c_i is drawn from a different scheme. Each processor applies his algorithm in parallel to each of the entries in the vector, verifying each coupon independently and applying the appropriate combining operation to each c_i.

A complication is that an adversary given a vector of coupons might choose to propagate only some of the c_i, while replacing others with dummy coupons. We can enable the receiver to detect when it has received a complete message by representing each bit x_i by two coupons: c_{2i-1} (for $x_i = 0$) and c_{2i} (for $x_i = 1$). A signal coupon in either position tells the receiver both the value of the bit and that the receiver has successfully received it.

Alas, we must construct and run $\Omega(m)$ blind coupon schemes in parallel to transmit m bits.

Better Construction. Some additional improvements in efficiency are possible. As before, our group structure is (U, G, D). Suppose our cyclic group G has order $n_0 p_1 \cdots p_m$, where p_i are distinct primes. Let D be the subgroup of G of order n_0.

An m-bit message $x = x_1 \dots x_m$ is encoded by a coupon $y \in G$, whose order is divisible by $\prod_{i\,:\,x_i=1} p_i$. For all i, we can find an element $g_i \in G$ of order $n_0 p_i$. We can thus let $y = g_1^{r_1 x_1} \cdots g_m^{r_m x_m}$ for random $r_1, \dots, r_m \in \{0, 1, \dots, 2^{2k} - 1\}$.

When we combine two coupons y_1 and y_2, it is possible that the order of their combination $\mathcal{C}_{PK}(y_1, y_2)$ is less than the l.c.m. of their respective orders. However, if the primes p_i are sufficiently large, this is unlikely to happen.

In Section 3.3.1, n_0 is a product of two moderately large primes, while the other primes can be around 2^{80}. For the construction from Section 3.3.2, n_0 is prime, but every prime must be fairly large to counter elliptic curve factorization.

This technique allows us to transmit messages of quite restricted bandwidth. It remains an open problem whether some other tools can be used to achieve higher capacity without a linear blow-up in message size.

3.4 Application of the BCM

In this section, we show how the BCM can be used to spread an alert quietly and quickly throughout a network. We begin with a definition of the problem in Sections 3.4.1, and then present results on the security and performance of the mechanism in Sections 3.4.2 and 3.4.3.

To summarize these results briefly, we consider a very general message-passing model in which each node P_i has a "split brain," consisting of an **update algorithm** $\mathcal{U}_i$ that is responsible for transmitting and combining coupons, and a **supervisor algorithm** $\mathcal{S}_i$ that may insert a signal coupon into the system at some point. The nodes carry out these operations under the control of a PPT **attacker** $\mathcal{A}$ that can observe all the external operations of the nodes and may deliver any message to any node at any time, including messages of its own invention.

We show first that, assuming the BCM is secure, the attacker can neither detect nor forge alerts despite its total control over message traffic. This result holds no matter what update algorithm is used by each node; indeed, it holds even if the update half of each node colludes actively with the adversary. We then give examples of some simple strategies for spreading an alert quickly through the network with some mild constraints on the attacker's behavior.

3.4.1 Our Model

We now describe the model for our algorithms.

Basic Setting

We adopt a very general message-passing communications model, permitting an active adversary both control over the timing of delivery of messages between nodes and the ability to read, replace, and redirect messages at will. At the same time, we structure our model of a node to enforce the requirement that the node's visible behavior (*e.g.*, its choices of what other nodes to communicate with) is not affected by the type of coupons it is transmitting.

Processes

We assume that we have a collection of n nodes $P_1, P_2, \ldots, P_n$. Processes have "split brains": for each node P_i an **update algorithm** $\mathcal{U}_i$ handles communication with other nodes, while a **supervisor algorithm** $\mathcal{S}_i$ chooses when or if to send a signal coupon. This split enforces the requirement that the communication pattern does not depend on which type of coupon a node is sending.

We do not examine the behavior of the supervisor algorithm closely; instead, we assume only that it supplies a sequence of coupons $c_i^1, c_i^2, \ldots$ to the update algorithm $\mathcal{U}_i$. The supervisor algorithm $\mathcal{S}_i$ of regular nodes will intermittently supply a sequence of dummy coupons. Meanwhile, $\mathcal{S}_i$ of sentinel nodes will supply dummy coupons until it detects the intruder's presence, at which point it will switch to dispensing signal coupons. We assume that the sequence does not depend on the execution of the rest of the protocol. For convenience, we write $\hat{c}_i^t$ for the indicator variable that c_i^t is a signal coupon; that is, we write $\hat{c}_i^t = 0$ if at step t of execution the coupon supplied by the supervisor algorithm of node P_i is a dummy coupon and $\hat{c}_i^t = 1$ if it is signal.

The inputs to update algorithm $\mathcal{U}_i$ at step t consist of (a) the sequence of sets of messages received at steps 1 through t; (b) the sequence of sets of messages sent at steps 1 through $t-1$; and (c) the coupon c_i^t supplied by $\mathcal{S}_i$ at time t. The output of $\mathcal{U}_i$ is a set of messages to be sent at step t. Each message is of the form (s, r, m, c) where s is the identity of the sender, r is the intended recipient, m is an arbitrary string, and c is a coupon. To simplify the model, we do not keep track of a separate process state, because any such state can easily be recomputed from the message history.

The update algorithms have access to the public key PK of the blind coupon mechanism. We assume that they can apply the verification algorithm $\mathcal{V}_{PK}$ and the combining algorithm $\mathcal{C}_{PK}$ in computing outgoing messages. To spread alerts, a typical update algorithm will discard any coupons from incoming messages or the supervisor algorithm that are rejected by $\mathcal{V}_{PK}$, and forward to a carefully-chosen set of recipients coupons obtained by combining all unrejected coupons so far in some order using $\mathcal{C}_{PK}$. It may also use additional information in messages to manage spreading of alerts, and this additional information may also depend on the values of the coupons it has seen.

Attacker

The PPT attacker algorithm $\mathcal{A}$ controls the timing and content of delivered messages. The input to the attacker is a partial execution, where the t-th step of an execution is described by a tuple (i_t, R_t, S_t) where i_t is a node identity, R_t is the set of messages received by P_{i_t} at that step, and S_t is the set of messages sent by P_{i_t} at that step. The output of $\mathcal{A}$ is a choice of which node $P_{i_{t+1}}$ executes the next step and what set of messages R_{t+1} it receives. The attacker also has access to the public key PK and can use the verification and combining algorithms $\mathcal{V}_{PK}$ and $\mathcal{C}_{PK}$ as subroutines.

An execution is constructed by an interactive protocol which alternates between the attacker choosing a node $P_{i_{t+1}}$ and a set of received messages R_{t+1} and the node's update algorithm $\mathcal{U}_i$ computing a set of messages S_{t+1} to send. Given particular public and secret keys, PK and SK, adversary $\mathcal{A}$, update algorithms $\mathcal{U}_i$, and supervisor inputs $\hat{c}_i^t$ for steps $t = 1, \ldots, T$, there exists a corresponding probability distribution $\Xi(PK, SK, \mathcal{A}, \{\mathcal{U}_i\}, \{\hat{c}_i^t\})$ on executions.

Note that traditional classes of process faults are easily simulated by an attacker defined in this way: a Byzantine node, for example, can be simulated by replacing all of its outgoing messages in transit. The attacker also has full power to violate any assumptions about synchrony, timely delivery, or reliable message transmission that the algorithm makes. We will show that such violations do not affect the security guarantees derived from the blind coupon mechanism; however, any performance guarantees on alert-spreading will require imposing restrictions on the attacker's behavior.

Problem

The problem is simple: at an opportune time, the sentinel nodes wish to propagate an alert (signal coupons) to all other nodes. We want to prevent the attacker (except with negligible probability) from (a) identifying the presence or source of signal coupons; (b) causing the nodes to spread signal coupons even though no supervisor algorithm supplied one; (c) preventing the spread of signal coupons to potential recipients.

3.4.2 Security

Let us begin with the security properties we want our alert-spreading mechanism to have.

Definition 7 *A set of update algorithms $\{\mathcal{U}_i\}$ is* ***secure*** *if, for any adversary algorithm $\mathcal{A}$, and any $T = poly(k)$, we have:*

1. ***Undetectability**: Given two distributions on executions, one in which no signal coupons are injected by supervisors and one in which some are, the adversary cannot distinguish between them with probability greater than 1/2. Formally, let $\hat{c}_i^{0,t} = 0$ for all i, t and let $\hat{c}_i^{1,t}$ be arbitrary. Then for any PPT algorithm $\mathcal{D}$,*

$$\left| \Pr \left[b = b' \;\middle|\; \begin{array}{c} (PK, SK, d, s) \leftarrow \mathcal{G}(1^k); \\ b \xleftarrow{\$} \{0,1\}; \\ \xi \xleftarrow{\$} \Xi\left(PK, SK, \mathcal{A}, \{\mathcal{U}_i\}, \{\hat{c}_i^{b,t}\}\right); \\ b' \leftarrow \mathcal{D}(1^k, PK, d, \{\hat{c}_i^{1,t}\}, \xi) \end{array} \right] - \frac{1}{2} \right| \leq negl(k).$$

2. ***Unforgeability**: The adversary cannot cause any process to transmit a signal coupon unless one is supplied by a supervisor. Formally, if $\hat{c}_i^t = 0$ for all i, t, then there is no PPT algorithm $\mathcal{A}$ such that*

$$\Pr \left[\exists (s, r, m, c) \in \xi \wedge (c \in S_{SK}) \;\middle|\; \begin{array}{c} (PK, SK, d, s) \leftarrow \mathcal{G}(1^k); \\ \xi \xleftarrow{\$} \Xi\left(PK, SK, \mathcal{A}, \{\mathcal{U}_i\}, \{\hat{c}_i^t\}\right); \end{array} \right] \leq negl(k).$$

Security of the alert-spreading mechanism follows immediately from the security of the underlying blind coupon mechanism. The essential idea behind undetectability is that because neither the adversary nor the update algorithms can distinguish between dummy and signal coupons distributed by the supervisor algorithms, there is no test that can detect their presence or absence. For unforgeability, the inability of the adversary and update algorithms to generate a signal coupon follows immediately from the unforgeability property of the BCM.

Theorem 8 *An alert-spreading mechanism is secure if the underlying blind coupon mechanism is secure.*

Proof: We show first undetectability and then unforgeability.

Undetectability. Suppose that the alert-spreading mechanism does not satisfy undetectability, *i.e.* that there exists a set of update algorithms $\{\mathcal{U}_i\}$, an adversary $\mathcal{A}$, and pattern $\{\hat{c}_i^{1,t}\}$ of signal coupons that can be distinguished from only dummy coupons by some PPT algorithm $\mathcal{D}$ with non-negligible probability.

Let us use this fact to construct a PPT algorithm $\mathcal{B}$ that violates indistinguishability. Let y be the coupon input to $\mathcal{B}$. Then $\mathcal{B}$ will simulate an execution ξ of the alert-spreading protocol by simulating the adversary $\mathcal{A}$ and the appropriate update algorithm $\mathcal{U}_i$ at each step. The only components of the protocol that $\mathcal{B}$ cannot simulate directly are the supervisor algorithms $\mathcal{S}_i$, because $\mathcal{B}$ does not have access to signal coupons provided to the supervisor algorithms of sentinel nodes. But here $\mathcal{B}$ lets $c_i^t = \mathcal{C}(d, d)$ when $\hat{c}_i^{1,t} = 0$ and lets $c_i^t = \mathcal{C}(y, y)$ when $\hat{c}_i^{1,t} = 1$. By the blinding property of the BCM, if $y \in D_{SK}$, then all coupons c_i^t will be statistically indistinguishable from uniformly random dummy coupons, giving a distribution on executions that is itself statistically indistinguishable from $\Xi\left(PK, SK, \mathcal{A}, \{\mathcal{U}_i\}, \{\hat{c}_i^{0,t}\}\right)$. If instead $y \in S_{SK}$, then c_i^t will be such that the resulting distribution on executions will be statistically indistinguishable from $\Xi\left(PK, SK, \mathcal{A}, \{\mathcal{U}_i\}, \{\hat{c}_i^{1,t}\}\right)$. It follows from the indistinguishability property of the BCM that no PPT algorithm $\mathcal{D}$ can distinguish between these two distributions with probability greater than $1/2 + negl(k)$.

Unforgeability. The proof of unforgeability is similar. Suppose that there is some adversary and a set of update functions that between them can, with non-negligible probability, generate a signal coupon given only dummy coupons from the supervisor algorithms. Then a PPT algorithm $\mathcal{B}$ that simulates an execution of this system and returns a coupon obtained by combining all valid coupons sent during the execution forges a signal coupon with non-negligible probability, contradicting the unforgeability property of the BCM.

■

3.4.3 Performance

It is not enough that the attacker cannot detect or forge alerts: a mechanism that used no messages at all could ensure that. In addition, we want to make some guarantee that if an alert is injected into the system, it eventually spreads to all non-faulty nodes. To do so requires both specifying a particular strategy for the nodes' update algorithms and placing restrictions on the attacker's ability to discard messages. We give two simple examples of how the blind coupon mechanism might be used in practice. More sophisticated models can also be used; the important thing is that security is guaranteed as long as the spread of coupons is uncorrelated with their contents.

A SYNCHRONOUS FLOODING MODEL. Consider a **communication graph** with an edge from each node to each other node that it can communicate to. Suppose that at step t, node P_i's update algorithm (a) discards all invalid incoming coupons; (b) combines any remaining coupons with its previous sent coupons and c_i^t; and (c) sends the result to all of its neighbors in the communication graph. Suppose further that nodes are divided into faulty and non-faulty nodes (by arbitrary choice of the attacker), and that every message sent by a non-faulty node to another non-faulty node is delivered intact by the attacker within at most one time unit. If the communication graph after deletion of faulty nodes is strongly connected, every node receives a signal coupon in at most Δ steps after a signal coupon is injected, where Δ is the diameter of the subgraph of non-faulty nodes.

A Simple Epidemic Model. In this model, the communication graph is complete, and at each step a randomly-chosen node chooses a random node to receive its coupon (which does so immediately). The behavior of a node receiving a message is the same as in the synchronous case. Then the number of interactions from the injection of the first signal coupon until all nodes possess a signal coupon is easily seen to be $O(n \log n)$. Formally:

Theorem 9 *Consider an execution ζ with n nodes of which $b < n$ are Byzantine, and suppose that some sentinel node begins sending a signal at the first step. Let the schedule be determined by choosing pairs of nodes for each step uniformly at random. Then all non-faulty nodes update their state to a signal coupon within expected $O(\frac{n^2 \log n}{n-b})$ steps.*

Proof: First observe that we can assume $b < n-1$, or else the unique non-faulty node possesses the alert at time 1.

Define a node as "alerted" if its state is a signal coupon, and let k be the number of alerted nodes. If the next step pairs an alerted, non-faulty node with a non-alerted, non-faulty node, which occurs with probability $\frac{k(n-b-k)}{n(n-1)}$, the number of alerted nodes rises to $k+1$. The expected time until this event occurs is at most $\frac{n(n-1)}{k(n-b)} < \frac{n^2}{k(n-b-k)}$.

The expected time until all non-faulty nodes are alerted is thus at most

$$\begin{aligned}
\sum_{k=1}^{n-b-1} \frac{n^2}{k(n-b-k)} &\leq n^2 \left(\sum_{k=1}^{\left\lceil \frac{n-b-1}{2} \right\rceil} \frac{1}{k\left(\frac{n-b-1}{2}\right)} + \sum_{k=\left\lfloor \frac{n-b-1}{2} \right\rfloor}^{n-b-1} \frac{1}{\left(\frac{n-b-1}{2}\right)(n-b-k)} \right) \\
&\leq 2n^2 \frac{2}{n-b-1} \sum_{k=1}^{\left\lceil \frac{n-b-1}{2} \right\rceil} \frac{1}{k} \\
&= \frac{4n^2}{n-b-1} H\left(\left\lceil \frac{n-b-1}{2} \right\rceil \right) \\
&= O\left(\frac{n^2 \log n}{n-b} \right).
\end{aligned}$$

■

If b is any constant fraction of n, the bound becomes simply $O(n \log n)$.

3.5 Generic Security of the Subgroup Escape Problem

We prove that the subgroup escape problem is hard in the generic group model [Sho97] when the representation set is much larger than the group.

Theorem 10 *Let D be a subgroup of $G \subseteq U$. Let g be a generator of D. Let $\mathcal{A}$ be a generic algorithm that solves the subgroup escape problem. If $\mathcal{A}$ makes at most q queries to the group oracle, then*

$$\Pr\left[y \in G \setminus D \;\middle|\; \mathcal{A}(1^k, \sigma(g)) = \sigma(y) \right] \leq \frac{q(|G| - |D|)}{(|U| - q)}.$$

Proof: The algorithm can only get information about σ through the group oracle. If the input to the oracle is two elements known to be in $\sigma(D)$, then the adversary learns a new element in $\sigma(D)$.

To have any chance of finding an element of $\sigma(G \setminus D)$, the adversary must use the group oracle to test elements that are not known to be in $\sigma(D)$.

Suppose that after i queries, the adversary knows a elements in $\sigma(D)$ and b elements of $U \setminus \sigma(G)$ $(a + b \leq i)$. For any z outside the set of tested elements, the probability that $z \in \sigma(G \setminus D)$ is exactly $(|G| - |D|)/(|U| - b)$ (note that it is independent of a).

Therefore, the probability that the adversary discovers an element in $\sigma(G \setminus D)$ with $i+1$ query is at most $(|G| - |D|)/(|U| - i)$. For up to q queries, the probability that at least one of the tested elements are in $\sigma(G \setminus D)$ is at most

$$\sum_{i=1}^{q} \frac{|G| - |D|}{|U| - i} \leq q \cdot \frac{|G| - |D|}{|U| - q}.$$

For a sufficiently large universe U, this probability is negligible. ■

3.6 Related Work

Our motivating example of spreading alerts is related to the problem of anonymous communication. Below, we describe known mechanisms for anonymous communication, and contrast their properties with what can be obtained from the blind coupon mechanism. We then discuss literature on elliptic curves over a ring, which are used in our constructions.

3.6.1 Anonymous Communication

Two basic tools for anonymous message transmission are DC-nets ("dining-cryptographers" nets) [Cha88, GJ04] and mix-nets [Cha81]. These tools try to conceal who the message sender and recipient are from an adversary that can monitor all network traffic. While our algorithms likewise aim to hide who the signal's originators are, they are much less vulnerable to disruption by an active adversary that can delay or alter messages, and they can also hide the fact that a signal is being spread through the network.

DC-nets enable one participant to anonymously broadcast a message to others by applying a dining cryptographers protocol. A disadvantage of DC-nets for unstructured systems like peer-to-peer networks is that they require substantial setup and key management, and are vulnerable to jamming. In contrast, the initialization of our alert-spreading application involves distributing only a public key used for verification to non-sentinel nodes and requires only a single secret key shared between the sentinels and the receiver, jamming is prevented by the verification algorithm, and outsiders can participate in the alert-spreading (although they cannot initiate an alert), which further helps disguise the true source. As the signal percolates across the network, all nodes change to an alert state, further confounding the identification of an alert's primary source even if a secret key becomes compromised.

The problem of hiding the communication pattern in the network was first addressed by Chaum [Cha81], who introduced the concept of a **mix**, which shuffles messages and routes them, thereby confusing traffic analysis. This basic scheme was later extended in [SRG00,SGR98]. A further refinement is a **mix-net** [Abe99,Jak99, Jak98], in which a message is routed through multiple trusted mix nodes, which try to hide correlation between incoming and outgoing messages. Our mechanism is more efficient and produces much stronger security while avoiding the need for trusted nodes; however, we can only send very small messages.

Beimel and Dolev's [BD01] proposed the concept of buses, which hide the message's route amidst dummy traffic. They assume a synchronous system and a passive adversary. In contrast, we assume both an asynchronous system and very powerful adversary, who in addition to monitoring the network traffic controls the timing and content of delivered messages.

3.6.2 Elliptic Curves over a Ring

One of our BCM constructions is based on elliptic curves over the ring $\mathbb{Z}_n$, where $n = pq$ is a product of primes. Elliptic curves over $\mathbb{Z}_n$ have been studied for nearly twenty years and are used, *inter alia*, in Lenstra's integer factoring algorithm [HWL87] and the Goldwasser-Kilian primality testing algorithm [GK99]. Other works [Dem93, KMOV92, OU98] exported some factoring-based cryptosystems (RSA [RSA78], Rabin [Rab79]) to the elliptic curve setting in hopes of avoiding some of the standard attacks. The security of our BCM relies on a special feature of the group of points on elliptic curves modulo a composite: It is difficult to find new elements of the group except by using the group operation on previously known elements. This problem has been noted many times in the literature, but was previously considered a nuisance rather than a cryptographic property. In particular, Lenstra [HWL87] chose the curve and the point at the same time, while Demytko [Dem93] used twists and x-coordinate only computations to compute on the curve without y-coordinates. To the best of our knowledge, this problem's potential use in cryptographic constructions was first noted in [Gjø04].

Our other BCM construction uses bilinear groups of composite order. Similar groups have recently been used by Boneh-Goh-Nissim [BGN05] to construct a cryptosystem with interesting homomorphic properties. Their proof of security also relies on the difficulty of the subgroup membership problem, which instills further confidence in our hardness assumptions.

3.6.3 Epidemic Algorithms

Our alert mechanism belongs to the class of epidemic algorithms (also called gossip protocols) introduced in [DGH+87]. In these algorithms, each process chooses to partner processes with which to communicate randomly. The drawback of gossip protocols is the number of messages they send, which is in principle unbounded if there is no way for the participants to detect when all information has been fully distributed.

Chapter 4

Verifiable Random Function*

We now consider another useful tool, called a **verifiable random function** (VRF). Our construction is simple and efficient. It has already been used in several practical applications [CHL05, CHYC05]. Furthermore, as we show in Chapter 5, it can be made distributed and proactive.

4.1 Overview

The notion of a **verifiable random function** (VRF) was introduced by Micali, Rabin, and Vadhan [MRV99]. Essentially, a VRF is a commitment to an exponential number of random-looking bits. Given an input value x, the knowledge of the secret key SK enables computing the function value $y = F_{SK}(x)$ together with the proof of correctness π_x. This proof convinces every verifier that the value $y = F_{SK}(x)$ is indeed correct with respect to the public key of the VRF.

Prior VRF constructions [MRV99, Lys02] (with the exception of [Dod03]) first construct a **verifiable unpredictable function** (VUF), whose output is hard to predict but does not necessarily look random. Then, they use an inefficient Goldreich-

*This chapter describes joint work with Yevgeniy Dodis and was originally reported in [DY05].

Levin transformation[1] to convert a VUF into a VRF, losing a factor in security along the way. Before the VRF value can be computed, inputs have to be mapped to either codewords of an error-correcting code or primes in a complicated fashion. Lastly, the size of proofs and keys of VRFs in [Lys02, Dod03] is linear in the input size. Meanwhile, the VRF of Micali-Rabin-Vadhan [MRV99] operates over the multiplicative group $\mathbb{Z}_n^*$ which has to be very large to achieve reasonable security.

In this chapter, we construct a simple VRF on groups equipped with bilinear maps. We forego using the Goldreich-Levin transformation, thereby saving several factors in security. The VRF's inputs need not be primes or codewords. For small inputs, our VRF has constant size proofs and keys. By utilizing a collision-resistant hash function, our VRF can also be used with inputs of arbitrary size. Our VRF construction can be instantiated with elliptic groups of reasonable size, which makes it quite practical.

Overview of Our Constructions. We begin by considering a signature due to Boneh and Boyen [BB04b]. On input x and a secret key SK, the signature is $\textsc{Sign}_{SK}(x) = g^{1/(x+SK)}$. Boneh and Boyen proved this signature to be existentially unforgeable against **non-adaptive** adversaries. By restricting inputs to have slightly superlogarithmic size (in security parameter), we are able to prove security against **adaptive** adversaries. Our proof is thus more involved, but necessarily less tight than the proof of Boneh and Boyen. We obtain a VUF, which is secure for small inputs. This VUF can then be converted into a VRF using the approach of prior works [MRV99, Lys02]. Specifically, we could use the Goldreich-Levin transformation [GL89] to convert it into a VRF with output size 1, amplify the output size to

[1]Generally, let $F_{SK} : \{0,1\}^l \mapsto \{0,1\}^l$ be a strong one-way function. Let $G_{SK} : \{0,1\}^{2l} \mapsto \{0,1\}^{2l}$ be defined by $G_{SK}(x,r) = (r, F_{SK}(x) \odot r)$, where $\odot$ denotes the inner dot product. Then, Goldreich and Levin showed that it is hard to guess the bit given $F_{SK}(x)$ and r [GL89].

match the size of the input, and then follow a tree-based construction to get a VRF with arbitrary input size. Needless, to say this is rather inefficient.

Instead, we prefer to construct a VRF directly, saving several factors in security. On input x and a secret key SK, our VRF computes $(F_{SK}(x), \pi(x))$, where $F_{SK}(x) = e(g,g)^{1/(x+SK)}$ is the VRF value and $\pi(x) = g^{1/(x+SK)}$ is the proof of correctness. We can apply a collision-resistant hash function to large inputs to transform our VRF into a VRF with unrestricted input length. By making the group size sufficiently large, we can construct a VRF with inputs of size roughly 160 bits, which is the length of SHA-1 digests. In theory, we do not have to assume existence of collision-resistant hash functions, and can instead apply a variant of a generic tree transformation to amplify the input size. Even though VRF's keys and proofs no longer have constant size, they are still significantly shorter than in other construction [MRV99, Dod03, Lys02].

Our proofs of security rely on two assumptions:

– **q-Diffie-Hellman inversion assumption** (q-DHI) states that no efficient algorithm can compute $g^{1/x}$ on input $\left(g, g^x, \ldots, g^{(x^q)}\right)$ [MSK02];

– **q-decisional bilinear Diffie-Hellman inversion assumption** (q-DBDHI) states that no efficient algorithm can distinguish $e(g,g)^{1/x}$ from random even after seeing $\left(g, g^x, \ldots, g^{(x^q)}\right)$ [BB04a].

Chapter Organization. The chapter proceeds as follows. First, in Section 4.2, we formalize the notions of a VRF and a VUF. We describe problems assumed to be hard in our security proofs (q-DHI and q-DBDHI) in Section 4.3. We give our constructions and analyze their efficiency in Section 4.4. In Section 4.5, we sketch how to make our VRF distributed and proactive. We describe several applications of our results in Section 4.6. Finally, in Section 4.7, we analyze the q-DBDHI assumption in the generic group model. We show that if the adversary can distinguish $e(g,g)^{1/x}$

from random, then he will need to perform (at least) $\sqrt{\epsilon p/q}$ group operations in a generic group of size p.

4.2 Definitions

Before presenting our results, we review some basic definitions and assumptions.

Let k be a security parameter as before. Let $a : \mathbb{N} \mapsto \mathbb{N} \cup \{*\}$ and $b : \mathbb{N} \mapsto \mathbb{N}$ be any functions for which $a(k)$ and $b(k)$ are computable in poly(k) time (except when a takes the value $*$).[2]

Intuitively, a **verifiable random function (VRF)** behaves like a pseudorandom function, but also provides proofs of its outputs' correctness.

Definition 11 *A function family $F_{(\cdot)}(\cdot) : \{0,1\}^{a(k)} \mapsto \{0,1\}^{b(k)}$ is a family of VRFs if there exists a PPT algorithm* $\textsc{Gen}$ *and deterministic algorithms* $\textsc{Prove}$ *and* $\textsc{Ver}$ *such that $\textsc{Gen}(1^k)$ outputs a pair of keys (PK, SK); $\textsc{Prove}_{SK}(x)$ computes $\big(F_{SK}(x), \pi_{SK}(x)\big)$, where $\pi_{SK}(x)$ is the proof of correctness; and $\textsc{Ver}_{PK}(x, y, \pi)$ verifies that $y = F_{SK}(x)$ using the proof π. Formally, we require:*

1. ***Uniqueness**: no values $(PK, x, y_1, y_2, \pi_1, \pi_2)$ can satisfy $\textsc{Ver}_{PK}(x, y_1, \pi_1) = \textsc{Ver}_{PK}(x, y_2, \pi_2)$ when $y_1 \neq y_2$.*

2. ***Provability**: if $(y, \pi) = \textsc{Prove}_{SK}(x)$, then $\textsc{Ver}_{PK}(x, y, \pi) = 1$.*

3. ***Pseudorandomness**: for any PPT algorithm $\mathcal{A} = (A_1, A_2)$, who does not*

[2]When $a(k)$ takes the value of $*$, it means the VRF is defined for inputs of all length.

query its oracle on x (see below),

$$\Pr\left[\begin{array}{c|c} b = b' & \begin{array}{c} (PK, SK) \leftarrow \textsc{Gen}(1^k); (x, st) \leftarrow A_1^{\textsc{Prove}(\cdot)}(PK); \\ y_0 = F_{SK}(x); y_1 \leftarrow \{0,1\}^{b(k)}; \\ b \leftarrow \{0,1\}; b' \leftarrow A_2^{\textsc{Prove}(\cdot)}(y_b, st) \end{array} \end{array}\right] \leq \frac{1}{2} + negl(k)$$

A **verifiable unpredictable function (VUF)** is a close relative of a VRF. Essentially, it is a signature scheme, whose verification algorithm accepts at most one signature for every public key and message.

Definition 12 *A function family $F_{(\cdot)}(\cdot) : \{0,1\}^{a(k)} \mapsto \{0,1\}^{b(k)}$ is a family of VUFs, if it satisfies the same syntax, uniqueness and provability properties of the VRFs, except the pseudorandomness property is replaced by the following weaker property:*

*3' . **Unpredictability**: for any PPT algorithm $\mathcal{A}$, who does not query its oracle on x (see below),*

$$\Pr\left[\begin{array}{c|c} y = F_{SK}(x) & (PK, SK) \leftarrow \textsc{Gen}(1^k); (x, y) \leftarrow \mathcal{A}^{\textsc{Prove}(\cdot)}(PK) \end{array}\right] \leq negl(k)$$

For exact security bounds, we will occasionally say that $F_{(\cdot)}(\cdot)$ is an $(s'(k), \epsilon'(k))$ secure VRF (*resp.*, VUF) if no adversary $\mathcal{A}$, running in time $s'(k)$, can break the pseudorandomness (*resp.*, unpredictability) property with $\epsilon'(k)$ advantage.

4.3 Complexity Assumptions

We now state the hardness assumptions on which our constructions are based. In what follows, we let $\mathbb{G}$ be a bilinear group of prime order p, and let g be its generator.

4.3.1 Diffie-Hellman Inversion Assumption

Our VUF construction relies on the Diffie-Hellman inversion (DHI) assumption, which was originally proposed in [MSK02].

Definition 13 *The* q-***DHI problem*** *in* $\mathbb{G}$ *asks: given the tuple* $\left(g, g^x, \ldots, g^{(x^q)}\right) \in (\mathbb{G}^*)^{q+1}$ *as input, compute* $g^{1/x}$.

An algorithm $\mathcal{A}$ has advantage ϵ in solving q-DHI in $\mathbb{G}$ if

$$\Pr\left[\mathcal{A}(g, g^x, \ldots, g^{(x^q)}) = g^{1/x}\right] \geq \epsilon,$$

where probability is taken over the coin tosses of $\mathcal{A}$ and the random choice of $x \in \mathbb{Z}_p^*$.[3] We say that (t, q, ϵ)-DHI assumption holds in $\mathbb{G}$ if, no t-time algorithm $\mathcal{A}$ has advantage at least ϵ in solving the q-DHI problem in $\mathbb{G}$.

Boneh and Boyen [BB04a] pointed out that the q-DHI assumption implies the $(q+1)$-generalized Diffie-Hellman assumption (GDH), on which many cryptographic constructions are based (*e.g.*, [NR97, BS02, STW96] as well as the VUF in [Lys02]). Therefore, security of our VUF rests on an equivalent complexity assumption to the one made before.

4.3.2 Decisional Bilinear Diffie-Hellman Inversion Assumption

In order to construct a VRF directly, we need to make a decisional bilinear Diffie-Hellman inversion assumption (DBDHI). It was previously used in [BB04a] to construct a selective-ID secure identity based encryption scheme.

[3]To simplify the notation, from now on, we assume that algorithms implicitly get a description of the bilinear group $(\mathbb{G}, \circ, p)$, on which they operate, as input.

Definition 14 *The q-**DBDHI problem** asks: given the tuple $\left(g, g^x, \ldots, g^{(x^q)}\right)$ as input, distinguish $e(g,g)^{1/x}$ from random.*

Formally, an algorithm $\mathcal{A}$ has advantage ϵ in solving the q-DBDHI problem if

$$\left| \Pr\left[\mathcal{A}(g, g^x, \ldots, g^{(x^q)}, e(g,g)^{1/x}) = 1\right] - \Pr\left[\mathcal{A}(g, g^x, \ldots, g^{(x^q)}, \Gamma) = 1\right] \right| \geq \epsilon,$$

where the probability is taken over the internal coin tosses of $\mathcal{A}$ and choices of $x \in \mathbb{Z}_p^*$ and $\Gamma \in \mathbb{G}_1$. We say that the (t, q, ϵ)-DBDHI assumption holds in $\mathbb{G}$ if no t-time algorithm $\mathcal{A}$ has advantage at least ϵ in solving the q-DBDHI problem in $\mathbb{G}$.

Clearly, q-DBDHI is a stronger assumption than q-DHI. To provide more confidence in its validity, we analyze this assumption in the generic group model in Section 4.7.

4.4 Our Constructions

In Section 4.4.1, we show that a signature scheme due to Boneh and Boyen [BB04b] is in fact a VUF for small inputs. We could then use a Goldreich-Levin hardcore bit to convert the resulting VUF into a VRF. However, the generic transformation is rather inefficient, so we choose to forego it. Instead, in Section 4.4.2, we construct our VRF directly for inputs of small size. We then show how to extend the VRF input size in Section 4.4.3. Finally, we evaluate our construction's efficiency in Section 4.4.4.

Fix input length $a(k)$, output length $b(k)$, and security $s(k)$. For notational convenience, we will usually omit the security parameter k, writing, for example, a or s, instead of $a(k)$ or $s(k)$. Let $\mathbb{G}$ $(|\mathbb{G}| = p)$ be a bilinear group, whose order p is a k-bit prime. Let g be a generator of $\mathbb{G}$. Throughout, we shall assume that messages can be encoded as elements of $\mathbb{Z}_p^*$.

4.4.1 A Verifiable Unpredictable Function

In order to build the intuition for our next proof, we first describe how to construct a simple VUF $(\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$, which is secure for small (superlogarithmic) inputs.

Algorithm $\textsc{Gen}(1^k)$: Chooses a secret $s \in_r \mathbb{Z}_p^*$ and sets the secret key to $SK = s$ and a public key to $PK = g^s$.

Algorithm $\textsc{Sign}_{SK}(x)$: Outputs the signature $\textsc{Sign}_{SK}(x) = g^{1/(x+SK)}$. Note that the proof is embedded in the output value so we do not need to include it explicitly.

Algorithm $\textsc{Ver}_{PK}(x, y)$: Outputs 1 if $e(g^x \cdot PK, y) = e(g, g)$; otherwise, outputs 0. Indeed, if the VRF value y was correctly computed, we have:

$$e(g^x \cdot PK, y) = e(g^x g^s, g^{1/(x+s)}) = e(g, g).$$

Boneh and Boyen [BB04b] proved this scheme to be existentially unforgeable against **non-adaptive adversaries** for inputs of arbitrary size. In our proof, we restrict inputs to have slightly superlogarithmic size in k (just like [MRV99] do); that is, we set $a(k) = \log s(k) = \Omega(\log k)$. This enables us to enumerate all possible messages in $s(k)$ time and to respond to adversary's queries **adaptively**. Further, the proof of [BB04b] is based on a q-strong Diffie-Hellman assumption (q-SDH), which is implied by a weaker q-DHI assumption used in our proof. Correspondingly, our proof is more involved but necessarily less tight than the proof of [BB04b].

Theorem 15 *Suppose the $(s(k), 2^{a(k)}, \epsilon(k))$-DHI assumption holds in a bilinear group $\mathbb{G}$ ($|\mathbb{G}| = p$). Let the input size be $a(k)$ and output size be $b(k) = \log_2 p$. Then $(\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ is a $(s'(k), \epsilon'(k))$ verifiable unpredictable function, where $s'(k) = s(k)/(2^{a(k)} \cdot poly(k))$ and $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$.*

Proof: It is easy to see that uniqueness and provability properties of Definition 12 are satisfied. We thus concentrate on residual unpredictability.

We shall use a shortcut and write $q = 2^{a(k)}$. Suppose there exists an adversary $\mathcal{A}$, running in time $s'(k)$, which guesses the value of the function at an unseen point with non-negligible probability $\epsilon'(k)$. We shall construct an algorithm $\mathcal{B}$ that by interacting with $\mathcal{A}$ breaks the q-DHI assumption with non-negligible probability.

Input to the reduction: Algorithm $\mathcal{B}$ is given a tuple $\left(g, g^{\alpha}, \ldots, g^{(\alpha^q)}\right) \in (\mathbb{G}^*)^{q+1}$, for some unknown $\alpha \in \mathbb{Z}_p^*$. Its goal is to compute $g^{1/\alpha}$.

Key generation: We guess that $\mathcal{A}$ will output a forgery on message $x_0 \in_r \{0,1\}^{a(k)}$. We are right with probability $1/2^{a(k)}$; error probability can be decreased by repeating the algorithm sufficiently many times. Let $\beta = \alpha - x_0$.[4] We don't know what β is because α is secret. However, we can use the Binomial Theorem to compute $\left(g^{\beta}, \ldots, g^{(\beta^q)}\right)$ from $\left(g^{\alpha}, \ldots, g^{(\alpha^q)}\right)$. Because $a(k) = \log(s(k))$, we can enumerate all possible inputs in $s(k)$ time. Let $f(z)$ be the polynomial

$$f(z) \quad = \prod_{w \in \{0,1\}^a, w \neq x_0} (z + w) = \sum_{j=0}^{q-1} c_j z^j \text{ (for some coefficients } c_0, \ldots, c_{q-1}).$$

We can compute

$$h = g^{f(\beta)} = \prod_{j=0}^{q-1} \left(g^{(\beta^j)}\right)^{c_j} \text{ and } h^{\beta} = \prod_{j=1}^{q} \left(g^{(\beta_j)}\right)^{c_{j-1}}.$$

Finally, we set h to be the generator and give $PK = h^{\beta}$ to $\mathcal{A}$. The secret key is $SK = \beta$, which we don't know ourselves.

Responding to oracle queries: Without loss of generality, we assume that $\mathcal{A}$

[4]For the sake of readability, we slightly abuse the notation. We should really have written $\beta = \alpha - \psi(x_0)$, where $\psi : \{0,1\}^{a(k)} \mapsto \mathbb{Z}_p^*$.

never repeats a query. Consider the ith query ($1 \leq i < q$) on message x_i. If $x_i = x_0$, then we fail. Otherwise, we must compute $\textsc{Sign}_{SK}(x_i) = h^{1/(x_i+\beta)}$. Let $f_i(z)$ be the polynomial

$$f_i(z) = f(z)/(z + x_i) = \sum_{j=0}^{q-2} d_j z^j \text{ (for some coefficients } d_0, \ldots, d_{q-2}).$$

We can compute

$$g^{f_i(\beta)} = \prod_{j=0}^{q-2} \left(g^{(\beta^j)}\right)^{d_j} = h^{1/(x_i+\beta)}$$

and return it as the signature.

Outputting the forgery: Eventually, $\mathcal{A}$ outputs a forgery (x^*, σ^*). If $x^* \neq x_0$, then our simulation failed. Because the signature is unique, we must have $\sigma^* = h^{1/(x_0+\beta)} = g^{f(\beta)/(x_0+\beta)}$. Compute

$$f(z)/(z + x_0) = \sum_{j=0}^{q-2} \gamma_j z^j + \frac{\gamma_{-1}}{z + x_0},$$

where $\gamma_{-1} \neq 0$. Hence,

$$\left(\sigma^* \cdot \prod_{j=0}^{q-2} \left(g^{(\beta^i)}\right)^{-\gamma_i}\right)^{1/\gamma_{-1}} = g^{1/(x_0+\beta)} = g^{1/\alpha}.$$

Let $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$ and $s'(k) = s(k)/(2^{a(k)} \cdot \text{poly}(k))$. To finish the proof, note that algorithm $\mathcal{B}$ succeeds with probability $\epsilon'(k)/2^{a(k)} = \epsilon(k)$. Its running time is dominated by answering oracle queries, and each query takes $(2^{a(k)} - 2) \cdot \text{poly}(k)$ time to answer. Therefore, $\mathcal{B}$ will run in roughly $s'(k) \cdot 2^{a(k)}\text{poly}(k) = s(k)$ time.

■

Remark: The security reduction of Theorem 15 is not tight. It allows to construct VUFs with input roughly $a(k) = \Omega(\log s(k))$. In theory, this means that the input size we can achieve might be only slightly superlogarithmic in k (similar to [MRV99]). First, it might be reasonable to assume subexponential hardness of the q-DHI assumption which will immediately allow one to support input of size $k^{\Omega(1)}$. Also, by utilizing a collision-resistant hash function, we will anyway only need to construct VUFs with relatively small input size such as 160 bits. Indeed, in Section 4.4.4, we show that our construction seems to yield a practical and secure VUF for inputs of arbitrary length already when $k = 1,000$ bits.

4.4.2 A Verifiable Random Function

Our main contribution is a direct construction of a verifiable random function from a slightly stronger q-DBDHI assumption. The VRF $(\textsc{Gen}, \textsc{Prove}, \textsc{Ver})$ is as follows.

Algorithm $\textsc{Gen}(1^k)$: Chooses a secret $s \in_r \mathbb{Z}_p^*$ and sets the secret key to $SK = s$ and the public key to $PK = g^s$.

Algorithm $\textsc{Prove}_{SK}(x)$: We let $\textsc{Prove}_{SK}(x) = \big(F_{SK}(x), \pi_{SK}(x)\big)$ where $F_{SK}(x) = e(g,g)^{1/(x+SK)}$ is the VRF output and $\pi_{SK}(x) = g^{1/(x+SK)}$ is the proof of correctness.

Algorithm $\textsc{Ver}_{PK}(x, y, \pi)$: To verify whether y was computed correctly, check if $e(g^x \cdot PK, \pi) = e(g,g)$ and whether $y = e(g, \pi)$. If both checks succeed, output 1; otherwise, output 0.

We can prove this scheme to be secure (in the sense of Definition 11) for small inputs (superlogarithmic in k). We then show how to convert it into a VRF with unrestricted input size.

Theorem 16 *Suppose the $(s(k), 2^{a(k)}, \epsilon(k))$-decisional BDHI assumption holds in a bilinear group $\mathbb{G}$ ($|\mathbb{G}| = p$). Let the input size be $a(k)$ and the output size be $b(k) = \log_2 p$. Then* $(\textsc{Gen}, \textsc{Prove}, \textsc{Ver})$*, as defined above, is a $(s'(k), \epsilon'(k))$ verifiable random function, where $s'(k) = s(k)/(2^{a(k)} \cdot poly(k))$ and $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$.*

Proof: It is trivial to show that uniqueness and provability properties of Definition 11 are satisfied. We thus concentrate on the pseudorandomness property.

We shall use $q = 2^{a(k)}$ as a shortcut. For sake of contradiction, suppose there exists an algorithm $\mathcal{A} = (A_1, A_2)$, which runs in time $s'(k)$, and can distinguish between $F_{SK}(x) = e(g,g)^{1/(x+s)}$ (for some x) and a random element in $\mathbb{G}_1$ with probability at least $1/2 + \epsilon'(k)$. We shall construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the q-DBDHI assumption in $\mathbb{G}$.

Input to the reduction: Algorithm $\mathcal{B}$ is given a tuple $(g, g^{\alpha}, \ldots, g^{(\alpha^q)}, \Gamma) \in (\mathbb{G}^*)^{q+1} \times \mathbb{G}_1$, where Γ is either $e(g,g)^{1/\alpha} \in \mathbb{G}_1$ or a random element in $\mathbb{G}_1$. Its goal is to output 1 if $\Gamma = e(g,g)^{1/\alpha}$ and 0 otherwise.

Key generation: We guess that $\mathcal{A}$ will choose to distinguish the VRF value on message $x_0 \in \{0,1\}^{a(k)}$. Let $\beta = \alpha - x_0$ (see footnote 4). We generate the public and private keys for algorithm $\mathcal{A}$ as in the proof of Theorem 15. Using the Binomial Theorem, we compute the tuple $\left(g^{\beta}, \ldots, g^{(\beta^q)}\right)$. We define

$$f(z) = \prod_{w \in \{0,1\}^a, w \neq x_0} (z + w) = \sum_{j=0}^{q-1} c_j z^j.$$

This enables us to compute the new base

$$h = g^{f(\beta)} = \prod_{j=0}^{q-1} \left(g^{(\beta^j)}\right)^{c_j}.$$

Finally, we give $PK = h^\beta = \prod_{j=1}^{q} \left(g^{(\beta^j)}\right)^{c_{j-1}}$ as the public key to $\mathcal{A}$. The secret key is $SK = \beta$, which we don't know.

Responding to oracle queries: Consider the ith query ($1 \leq i < q$) on message x_i. If $x_i = x_0$, we fail. Otherwise, we must respond with the corresponding proof $\pi_{SK}(x_i)$ and a VRF value $F_{SK}(x_i)$.

As in Theorem 15, we define

$$f_i(z) = f(z)/(z + x_i) = \sum_{j=0}^{q-2} d_j z^j \text{ (for some coefficients } d_0, \ldots, d_{q-2}\text{)}.$$

We can thus compute

$$\pi_{SK}(x_i) = \prod_{j=0}^{q-2} \left(g^{(\beta^j)}\right)^{d_j} = h^{1/(\beta + x_i)}$$

and

$$F_{SK}(x_i) = e(h, \pi_{SK}(x_i)) = e(h, h)^{1/(\beta + x_i)},$$

and return them to algorithm $\mathcal{A}$.

Challenge: Eventually, $\mathcal{A}$ outputs a message x^* on which it wants to be challenged. If $x^* \neq x_0$, then we fail. Otherwise, $\mathcal{A}$ claims to be able to distinguish $e(h, h)^{1/(\beta + x_0)} = e(h, h)^{1/\alpha}$ from a random element in $\mathbb{G}_1$. Recall that

$$f(z) = \sum_{i=0}^{q-1} c_i z^i.$$

Because $f(z)$ is not divisible by $(z + x_0)$, we have:

$$\begin{aligned} f'(z) &= f(z)/(z+x_0) - \frac{\gamma}{z+x_0} \\ &= \sum_{j=0}^{q-2} \gamma_j z^j \text{ (for some } \gamma \neq 0 \text{ and coefficients } \gamma_0, \ldots, \gamma_{q-2}). \end{aligned}$$

Let Γ_0 be

$$\begin{aligned} \Gamma_0 &= \left(\prod_{i=0}^{q-1} \prod_{j=0}^{q-2} e\left(g^{(\beta^i)}, g^{(\beta^j)}\right)^{c_i \gamma_j} \right) \cdot \left(\prod_{m=0}^{q-2} e\left(g, g^{(\beta^t)}\right)^{\gamma \gamma_m} \right) \\ &= e\left(g^{f(\beta)}, g^{f'(\beta)}\right) \cdot e\left(g^{\gamma}, g^{f'(\beta)}\right) \qquad (4.1) \\ &= e(g,g)^{(f(\beta)^2 - \gamma^2)/\alpha}. \end{aligned}$$

Set $\Gamma^* = \Gamma^{(\gamma^2)} \cdot \Gamma_0$. Notice that if $\Gamma = e(g,g)^{1/\alpha}$, then $\Gamma^* = e(g^{f(\beta)}, g^{f(\beta)/\alpha}) = e(h,h)^{1/\alpha}$. Meanwhile, if Γ is uniformly distributed, then so is Γ^*. We give Γ^* to algorithm $\mathcal{A}$.

Note: It may seem as though computing Γ_0 is very expensive. However, from Equation (4.1), we see that the computation only takes two bilinear map evaluations.

Guess: Algorithm $\mathcal{A}$ makes some more queries to which we respond as before. Finally, $\mathcal{A}$ outputs a guess $b \in \{0,1\}$. We return b as our guess as well.

The running time of the reduction is dominated by simulating oracle queries. Per every query, we must perform one bilinear map evaluation (this takes poly(k) time) and $(2^a - 2)$ multiplications and exponentiations (this takes $2^a \cdot \text{poly}(k)$ time). Because $\mathcal{A}$ can make at most $s'(k)$ queries, the running time of $\mathcal{B}$ is altogether $s'(k)(2^{a(k)} \cdot \text{poly}(k))$. The advantage of $\mathcal{B}$ in this experiment is $\epsilon'(k)/2^{a(k)}$. Setting

$s'(k) = s(k)/(2^{a(k)} \cdot \text{poly}(k))$ and $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$ completes the proof. ■

We can view the VRF output $F_{SK}(x) = h^{1/(x+SK)}$ as a pseudorandom function (PRF) on the group $\mathbb{G}_1$, whose generator is $h = e(g, g)$. By Theorem 16, this PRF is secure (for small inputs) under the decisional bilinear Diffie-Hellman inversion assumption.[5] To the best of our knowledge, this is the first PRF in the standard model, which does not process its inputs bit-by-bit. This PRF also admits efficient zero-knowledge proofs for statements of the form "$y = F_{SK}(x)$" and "$y \neq F_{SK}(x)$" given a commitment to the secret key SK (see [CL04] for an overview of these techniques).

4.4.3 Extending the Input Size

We constructed a VRF (Gen, Prove, Ver), which is provably secure for inputs of small size $a(k) = \Omega(\log(k))$. We now explain how to handle inputs of arbitrary size.

Hashing the Input.

Notice that if we have a VRF $\text{Prove}_{SK}(\cdot) : \{0,1\}^{a(k)} \mapsto \{0,1\}^{b(k)}$ and a collision-resistant hash function $H(\cdot) : \{0,1\}^* \mapsto \{0,1\}^{a(k)}$, then their composition $\text{Prove}_{SK}(H(\cdot)) : \{0,1\}^* \mapsto \{0,1\}^{b(k)}$ is trivially secure. Although our security reduction is relatively loose, we can make the size of a bilinear group large enough (we give exact numbers in Section 4.4.4) to have inputs of length roughly $a(k) = 160$ bits, the length of SHA-1 digests. Restriction to small inputs is therefore not limiting because we can always hash longer inputs.

[5] In fact, we no longer need to use bilinear maps for a PRF. So, we can replace the q-DBDHI assumption by a q-DDHI assumption: given the tuple $\left(h, h^x, \ldots, h^{(x^q)}\right) \in (\mathbb{G}_1^*)^{q+1}$ as input, distinguish $h^{1/x}$ from random [CHL05].

Tree Construction.

Although, we recommend using the previous construction (by making the group large enough), in theory, we could always use the (inefficient) generic tree construction to extend the input length. Then, we do not have to assume the existence of a collision-resistant hash function; having a universal hash function suffices.

We shall use the following proposition:

Proposition 17 ([MRV99]) *If there is a VRF with input length* $a(k)$*, output length 1, and security* $s(k)$*, then there is a VRF with unrestricted input length, output length 1 and security at least* $\min(s(k)^{1/5}, 2^{a(k)/5})$.

The construction first converts a VRF with output length 1 into a VRF with output length $(a-1)$. This transformation loses a factor of a in security. Because our VRF has output length much larger than 1, we can omit this step. Instead, we apply a universal hash function to VRF's output and let the VRF's value be the first $(a-1)$ bits of hash function's output (it is easily seen that these bits will be pseudo-random as well).

The rest of the transformation proceeds as usual. We construct a binary trie whose nodes are labeled with strings of length $(a-1)$. The root is labeled with 0^{a-1} and the children of node y are labeled with VRF values on inputs $(y \circ 0)$ and $(y \circ 1)$. Computing the VRF value on input $x \in \{0,1\}^*$ amounts to tracing a path through the trie to the leaf corresponding to x. The VRF value is the label of the leaf, and the proof of correctness is a tuple of VRF proofs—one proof per each node on the path traced by x.[6]

We also note that both of the aforementioned techniques can be used to convert the VUF in Section 4.4.1 into a VUF with unrestricted input length.

[6]The inputs have to be prefix-free for this tree construction to work. This can be accomplished using techniques of [MRV99].

4.4.4 Efficiency

We now compare the efficiency of our construction with that of prior VRF constructions. We fix inputs to be $a(k) = 160$ bits, the length of SHA-1 digests, and let $q = 2^{a(k)}$.

Our VRF.

According to Theorem 16, if $(s(k), q, \epsilon(k))$-DBDHI holds on $\mathbb{G}$, then our VRF is secure against adversaries running in time $s'(k) = s(k)/(2^{a(k)} \cdot \text{poly}(k))$ that have advantage $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$. To be generous, we instantiate $\epsilon'(k) = 2^{-80}$, $s'(k) = 2^{80}$, and $\text{poly}(k) = 2^{30}$. Then, we have: $\epsilon(k) = 2^{-240}$ and $s(k) = 2^{270}$. Suppose no better algorithm exists for breaking the q-DBDHI assumption than a generic group algorithm. Then, by Theorem 18 (which we prove in Section 4.7), for these security parameters a bilinear group must have size:

$$\begin{aligned} p &\geq \frac{2(s(k) + q + 3)^2 q}{\epsilon(k)} \\ &= \frac{2\left(2^{270} + 2^{160} + 3\right)^2 2^{160}}{2^{-240}} \\ &\approx 2^{940}. \end{aligned}$$

Therefore, making the group size be a 1,000 bit prime seems sufficient to guarantee security of the VRF that takes 160 bit inputs. Proofs and keys consist of a single group element and will roughly be 125 bytes each. We can generate such groups using the standard parameter generator of [BF01a].

VRF by Micali-Rabin-Vadhan [MRV99].

This VRF operates over a multiplicative group $\mathbb{Z}_n^*$, where $n = pq$ is a k-bit RSA modulus. The fastest general-purpose factoring algorithm today is the number field sieve [BLZ94]; it takes approximately $O\left(e^{1.9223(k^{1/3}(\log k)^{2/3})}\right)$ time to factor a k bit number. The RSA based VUF (not even a VRF) constructed in [MRV99] has security $s'(k) = s(k)/(2^{a(k)} \cdot \text{poly}(k))$ where $s(k)$ is hardness of RSA. Letting $s'(k) = 2^{80}$ and $\text{poly}(k) = 2^{30}$ as before, we obtain an RSA security lower bound $s(k) = 2^{80} \cdot (2^{160} \cdot 2^{30}) = 2^{270}$. Because RSA is only secure as long as we cannot factor n, to get 270 bits of security, we need n to be a k-bit number, where

$$1.9223k^{1/3}(\log k)^{2/3} = 270.$$

Hence, n must be at least $14,383$ bits long if we want to use this VUF on 160 bit inputs. After following the tree construction, proofs for 160 bit inputs will have size 280 kilobytes.

VRF by Dodis [Dod03] and VUF by Lysyanskaya [Lys02].

These constructions work on elliptic curve groups, whose size is usually a 160 bit prime. At the bare minimum, 160 bit messages yield keys and proofs of size $160 \cdot 160 = 25,600$ bits, which is about 3.2 kilobytes. In fact, they will probably have larger size due to use of error-correcting codes and other encoding expansions.

To summarize, none of the prior VRF constructions come close to the 1,000 bit proofs and keys of our construction. If our VRF is used with the generic tree construction, its keys and proofs consist of $|x|$ group elements (one group element per input bit) when the input is $x \in \{0,1\}^*$. This is less than the $|x|^2$ group elements ($|x|$ group elements per input bit) needed by the VRF of [Lys02].

4.5 Distributed VRF

We point out that our VUF/VRF constructions can be easily made distributed (or even proactive) as we describe in Chapter 5. Indeed, both of the constructions simply amount to a secure computation of the function $\pi_{SK}(x) = g^{1/(x+SK)}$ when the servers have shares of the secret SK. Because it is well known how to do multiparty addition, inversion, and exponentiation [BIB89, BoGW88], this extension follows immediately. We notice however that unlike the construction of Dodis [Dod03], our distributed VUF/VRF is interactive.

4.6 Applications of the VRF

Our PRF and VRF have useful algebraic properties, which make them attractive to use in distributed protocols. We describe two such protocols next.

4.6.1 Compact E-Cash

One way to make payments over the Internet is to use **e-cash** (electronic cash) [Cha82]. The simplest e-cash system involves a bank, which mints electronic coins; several shops, which sell merchandise in exchange for coins; and users, who **withdraw** coins from the bank and **spend** them at various shops. An electronic coin is just a bit string, so an unscrupulous user may copy the coin and try to spend it twice. In the offline scenario, shops accept coins autonomously and later **deposit** them to the bank. These coins will either be honored or will lead to the identification of the double-spender.

The best previously known e-cash schemes require $O(2^l \cdot k)$ bits to store a wallet of 2^l coins, where k is the security parameter. Recently, Camenisch *et al.* [CHL05] used our PRF to construct an efficient off-line e-cash system, where a wallet of 2^l coins can be stored in $O(l+k)$ bits (this is the same size as one coin in prior e-cash systems). Here is how it works.

Suppose user U has a secret key sk_U and a public key $pk_U = g^{sk_U}$. An electronic coin of user U is a tuple (sk_U, S_i, T_i), where S_i is coin's **serial number** and T_i is its **security tag**. In the withdrawal protocol, the user obtains a signature of (sk_U, s, t) under bank's public key. The idea is to use s and t as seeds to our PRF and generate multiple electronic coins $(sk_U, S_i = F_s(i), T_i = F_t(i))$ $(0 \le i \le 2^l - 1)$ without contacting the bank. In the spending protocol, the user sends to the shop a commitment C to (sk_U, s, t) and a proof π_1 that he has the bank's signature. For each coin, the shop chooses a random $R_i \in \mathbb{Z}_q$ and the user reveals the serial number S_i and the double-spending equation $D_i = pk_U(T_i)^{R_i} = g^{sk_U}(F_t(i))^{R_i}$,[7] along with a proof π_2 that values S_i and D_i correspond to commitment C.

Our PRF allows the user to efficiently compute zero-knowledge proofs (π_1, π_2) that the PRF values were derived correctly given commitments to PRF's input i and seeds s, t. Our PRF's output is one group element regardless of the input size, so electronic coins become very compact. Meanwhile, if we substituted other PRFs [Lys02, NR97] in place of ours, zero-knowledge proofs would be much harder to compute and the wallets would grow from $O(l+k)$ to (at least) $O(l \cdot k)$ bits in size.

4.6.2 Electronic Lottery

An **e-lottery** (electronic lottery) scheme allows to anonymously purchase lottery tickets and claim prizes over the Internet [MR02]. Typically, it includes one lottery

[7]If the same coin is spent twice, the double-spending equation can be solved for pk_U, which is sufficient to detect and identify dishonest users.

dealer and a large number of players. Each player bets on a single number, then the dealer executes a random drawing to determine the combination of winning numbers. Large sums of money are involved in each game, so security becomes a very important issue. Many existing e-lotteries require players to remain online during the drawing to ensure outcome's fairness. Clearly, this is not realistic in large-scale games involving millions of users.

Chow *et al.* [CHYC05] used our VRF to construct a practical e-lottery system, where lottery audit is performed offline. A bird's eye view of their protocol is as follows. Each player chooses his favorite number x. He engages in an interactive protocol with the dealer, at the end of which he obtains a ticket t. The lottery dealer holds the secret key SK of a VRF. Given a ticket t, he computes the value $y = F_{SK}(t)$ together with the proof $\pi = \pi_{SK}(t)$. The value y somehow determines if the user wins, while the proof π guarantees that the dealer cannot cheat.

Players need not stay online during the generation of winning tickets because VRF's proofs are non-interactively verifiable. The pseudo-randomness of VRF's outputs guarantees that no player can predict winning tickets better than guessing. Finally, using our VRF (as opposed to other VRFs [MRV99, Lys02, Dod03]) ensures lottery tickets and proofs are very small. In fact, the e-lottery is practical enough for players with mobile devices to participate in the game.

4.7 Generic Security of the q-DBDHI Assumption

In this section, we examine the q-DBDHI assumption in the generic group model of Shoup [Sho97]. We proceed to derive a lower bound on the running time of a generic adversary who breaks this assumption.

Theorem 18 *Let $\mathcal{A}$ be an algorithm that solves the q-DBDHI problem. Assume both*

$x \in \mathbb{Z}_p^*$ and the encoding functions σ, σ_1 are chosen at random. If $\mathcal{A}$ makes at most q_G queries to oracles computing the group action in $\mathbb{G}, \mathbb{G}_1$ and the bilinear mapping $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$, then

$$\left| \Pr\left[\begin{array}{c|c} \mathcal{A}\left(p, \sigma(1), \sigma(x), \ldots, \sigma(x^q), \right. & b \xleftarrow{\$} \{0,1\}; \\ \left. \sigma_1(\Gamma_0), \sigma_1(\Gamma_1)\right) = b & \Gamma_b \leftarrow 1/x;\ \Gamma_{1-b} \xleftarrow{\$} \mathbb{Z}_p^* \end{array} \right] - \frac{1}{2} \right| \le \frac{2(q_G + q + 3)^2 q}{p}.$$

Proof: Instead of letting $\mathcal{A}$ interact with the actual oracles, we play the following game.

We maintain two lists: $L = \{ (F_i, s_i) : i = 0, \ldots, t-1\}$ and $L' = \{ (F_i', s_i') : i = 0, \ldots, t'-1\}$. Here $s_i, s_i' \in \{0,1\}^*$ are encodings and $F_i, F_i' \in \mathbb{Z}_p[X, \Gamma_0, \Gamma_1]$ are multivariate polynomials in X, Γ_0, and Γ_1. The total length of lists at step $\tau \le q_G$ in the game must be

$$t + t' = \tau + q + 3. \tag{4.2}$$

In the beginning of the game, we initialize the lists to $F_0 = 1, F_1 = X, \ldots, F_q = X^q$ and $F_0' = \Gamma_0, F_1' = \Gamma_1$. The corresponding encodings are set to arbitrary distinct strings in $\{0,1\}^*$. The lists have length $t = q + 1$ and $t' = 2$.

We start the game by providing $\mathcal{A}$ with encodings $(s_0, \ldots, s_q, s_0')$. Algorithm $\mathcal{A}$ begins to issue oracle queries. We respond to them in the standard fashion:

Group action: Given a multiply/divide bit and two operands s_i and s_j ($0 \le i, j < t$), we compute $F_t = F_i \pm F_j$ accordingly. If $F_t = F_l$ for some $l < t$, we set $s_t = s_l$. Otherwise, we set s_t to a random string in $\{0,1\}^* \backslash \{s_0, \ldots, s_{t-1}\}$, and increment t by 1. Group action in G_1 is computed similarly, except we operate on list L'.

Bilinear pairing: Given two operands s_i and s_j ($0 \le i, j < t$), we compute the product $F_{t'} = F_i F_j$. If $F_{t'} = F_l$ for some $l < t'$, we set $s_{t'} = s_l$. Otherwise we set it to a random string in $\{0,1\}^*\backslash\{s_0,\ldots,s_{t'-1}\}$. We then increment t' by 1.

After making at most q_G queries, $\mathcal{A}$ halts with a guess $\hat{b} \in \{0,1\}$. We now choose $x, y \xleftarrow{\$} \mathbb{Z}_p^*$ and consider $\Gamma_b \leftarrow 1/x, \Gamma_{1-b} = y$ for both choices of b. Our simulation is perfect and reveals nothing to $\mathcal{A}$ about b unless the values that we chose for indeterminates give rise to some non-trivial equality relation. Specifically, algorithm $\mathcal{A}$ wins the game if for any $F_i \neq F_j$ or any $F'_i \neq F'_j$, either of these hold:

1. $F_i(x, 1/x, y) - F_j(x, 1/x, y) = 0$
2. $F_i(x, y, 1/x) - F_j(x, y, 1/x) = 0$
3. $F'_i(x, 1/x, y) - F'_j(x, 1/x, y) = 0$
4. $F'_i(x, y, 1/x) - F'_j(x, y, 1/x) = 0$

Notice that $\mathcal{A}$ can never engineer an encoding of an element whose corresponding polynomial would have a $1/X$ term unless he is explicitly given it. Therefore, we can only get a non-trivial equality relation as a result of numerical cancellation.

For all i, $\deg(F_i) \leq q$ and $\deg(F'_i) \leq 2q$. We can use the Schwartz-Zippel Theorem [Sch80] to bound the probability of a cancellation. It tells us that for all i, j, $\Pr[F_i - F_j = 0] \leq q/p$ and $\Pr[F'_i - F'_j = 0] \leq 2q/p$. Thus $\mathcal{A}$'s advantage is

$$\begin{aligned}
\epsilon &\leq 2 \cdot \left(\binom{t}{2} \frac{q}{p} + \binom{t'}{2} \frac{2q}{p} \right) \\
&< 2(q_G + q + 3)^2 \frac{q}{p} \quad \text{(plugging into (4.2))} \\
&= O\left(\frac{q_G^2 q + q^3}{p} \right).
\end{aligned}$$

■

The following corollary is immediate.

Corollary 19 *Any adversary that breaks the q-DBDHI assumption with probability* $\frac{1}{2} + \epsilon$ $(0 < \epsilon < 1/2)$ *in generic groups of order* p *such that* $q < o(\sqrt[3]{p})$ *requires* $\Omega(\sqrt{\epsilon p/q})$ *generic group operations.*

Chapter 5

Threshold Pseudo-Random Permutation*

In the previous chapter, we described a PRF and a VRF with constant-size outputs. A **pseudo-random permutation** (PRP) is effectively an invertible PRF. In this chapter, we use our PRF to construct the first reasonably efficient threshold and proactive pseudo-random permutation. The resulting protocol needs only $O(1)$ communication rounds. It tolerates up to $(n-1)/2$ of n dishonest servers in the semi-honest environment.

5.1 Overview

The security of cryptographic primitives relies on the owner of the secret key, who is expected to scrupulously protect it. Alas, we cannot always put all trust into a single party. In fact, the area of **threshold cryptography** deals exclusively with sharing the ability to perform cryptographic operations between a set of n servers [Des97]. A long line of research produced many distributed protocols that

*This chapter describes joint work with Yevgeniy Dodis and Moti Yung [DYY06].

are more efficient than generic multi-party solutions when used for public key encryption [Ped91,DJ01,Rab98], digital signatures [Des87,DF89,DF91,GJKR01], key generation [ACS02, BF01b, GJKR99], pseudo-random functions [Nie03, NPR99, CKPS01], and other applications. The extra security and increased availability of constructions justify the added complexity. The pseudo-random permutation is the only primitive that is still missing from this long list. Initial attempts by [BCF00] and [MSNWW05] gave a very basic sharing structure with many limitations and drawbacks.[1] The question of constructing a distributed PRP was yet left open. In this chapter, we resolve this problem.

We focus on implementing the Luby-Rackoff construction [LR88] as a method for building PRPs. It uses the **Feistel permutation** for function F (denoted $\bar{F}$), which sends a $2l$-bit string (x^L, x^R) to a $2l$-bit string $(x^R, x^L \oplus F(x^R))$. Luby and Rackoff showed that a composition of four Feistel permutations (denoted $\Psi(F_1, F_2, F_3, F_4) = \bar{F}_1 \circ \cdots \circ \bar{F}_4$) is a secure $2l$-bit PRP when F_i are independent l-bit PRFs. While a sequential composition of PRFs to build a sequential PRP is generic, there is a major technical difficulty in the distributed Luby-Rackoff construction. Particularly, the difficult part is that if one uses a PRF as an intermediate round function, then not just the secret key, but also the output needs to be kept distributed to assure the security of the entire Luby-Rackoff construction. At the same time, the computation needs to continue and compute on these shares, which means that we need to compute on shared inputs as well.

Overview of Our Construction. In our protocol, servers hold Shamir shares [Sha79] of secret keys SK_i to PRFs F_i used in the Luby-Rackoff (LR) construction of a PRP

[1]They showed how to build rather inefficient cascade ciphers $E_k(x) = g_{k_m}(\dots g_{k_2}(g_{k_1}(x)))$, where $g(\cdot)$ is itself a secure cipher, by sharing a sequence of keys in a special way. For τ-out-of-n sharing, the number of keys and composition layers is on the order of $\binom{n}{\tau}$, which is exponential for most τ.

$\Psi(F_1, F_2, F_3, F_4)$. The untrusted user who wants to compute the PRP's value broadcasts his input x to the servers. Servers somehow verify that the user is entitled to evaluate the PRP and engage in an interactive protocol, which terminates with shares of the PRP's value.

Our round functions F_i are based on our PRF from Chapter 4. We chose this PRF because it possesses useful algebraic properties and can be computed in $O(1)$ rounds. Recall that given an l-bit input $x = x_1 x_2 \ldots x_l$ (which can be viewed as an element of $\mathbb{Z}_Q$) and a secret key $SK \in \mathbb{Z}_Q$, the PRF value is $F_{SK}(x) = g^{1/(x+SK)}$. Here, g is the generator of a group in which the **decisional Diffie-Hellman inversion** (y-DDHI) problem is hard. The y-DDHI problem asks: "given $(g, g^x, \ldots, g^{(x^y)}, R)$ as input, to decide whether $R = g^{1/x}$ or not." It appears hard in a quadratic residues subgroup G_Q of $\mathbb{Z}_P^*$ ($P = 2Q + 1$) for sufficiently large primes P, Q.

In last chapter, we showed that $F_{SK}(\cdot)$ is secure only for inputs of small length $l = \omega(\log k)$, which makes it unsuitable for the LR construction, whose round functions must accept longer $l = \Theta(k)$ bit inputs (k is the security parameter). In this chapter, we assume subexponential hardness of the y-DDHI problem. This immediately allows us to support inputs of size $a = \Theta(k^\delta)$ for some small $\delta \approx 1/3$. We can shrink the input to the LR construction from $l = \Theta(k)$ bits down to $a = \Theta(k^\delta)$ bits using an ϵ-universal hash function $h_i(x) = (ix \bmod Q) \bmod 2^a$. We thus get a new PRF $F'_{i,SK}(x) = F_{SK}(h_i(x))$, which can be used in the (centralized) LR construction.

We distribute the LR construction using well-known multiparty tools of addition, multiplication, inversion, etc. [BIB89, BoGW88, ACS02]. We rely heavily on an $O(1)$ round protocol by Damgård *et al.* [DFK$^+$06], which computes shares of bits of $x \in \mathbb{Z}_P$ from shares of x. This protocol allows us to efficiently perform modular reduction, exponentiation, and truncation of shared values.

We can compute the PRF $F'_{i,SK}(x)$ with shared input x and keys (i, SK) as follows. Computing the ϵ-universal hash $h_i(x) = (ix \bmod Q) \bmod 2^a$ amounts to a single multiparty multiplication, followed by a call to Damgård *et al.*'s protocol to extract the trailing a bits. We can also distribute the computation of $F_{SK}(x) = g^{1/(x+SK)}$ because it is well-known how to do multiparty addition, inversion, and exponentiation. As a result, we obtain a sharing of $F'_{i,SK}(x)$, a random group element in G_Q, whereas we need a sharing of a random l-bit string. We can use a deterministic extractor $E(x) = (x^{(P+1)/4} \bmod P) \bmod 2^l$ to convert this group element into a random l-bit string. Computing this extractor distributively entails a single distributed exponentiation followed by a call to Damgard *et al.*'s protocol to extract l bits.

Armed with a protocol for computing the PRF $F'_{i,SK}(\cdot)$, we can distribute a single Feistel permutation, which maps (x^L, x^R) into $(x^R, x^L \oplus F'_{i,SK}(x^R))$. The only missing link is how to XOR shares of PRF's bits with shares of input's bits. Given shares of bits $b_1, b_2 \in \{0,1\}$, we can get a share of $b_1 \oplus b_2$ by distributively computing $(b_1 + b_2) \cdot (2 - (b_1 + b_2))$. We obtain a threshold PRP by iterating the distributed Feistel permutation four times, cross-feeding its outputs to inputs.

Chapter Organization. The remainder of the chapter is organized as follows. Section 5.2 reviews some tools from multiparty computation, which are used throughout this chapter. In Section 5.3, we give distributed protocols for evaluating our PRF from Chapter 4 and a PRF by Naor-Reingold [NR97] when the input and the secret key are shared. In Section 5.4, we present our distributed Luby-Rackoff protocol. Some practical applications of our construction appear in Section 5.5. In Section 5.6, we describe our new distributed exponentiation protocol. Finally, in Section 5.7, we analyze the generic hardness of the q-DDHI assumption.

5.2 Building Blocks

We denote protocols as follows: the term $[a]_j^P \leftarrow \texttt{PROTOCOL}([b]_j^P, c)$ means that server P_j executes the protocol $\texttt{PROTOCOL}$ with local input $[b]_j^P$ and public input c. As a result of the protocol, it gets back local output $[a]_j^P$. In all cases, the local inputs and outputs will be Shamir shares over the appropriate field.

Our protocols are secure against a static, honest-but-curious adversary who controls up to $\tau = \lfloor (n-1)/2 \rfloor$ (computationally bounded) servers. We prove security in the UC framework by Canetti [Can01]. In the honest-but-curious setting, privacy is preserved under non-concurrent modular composition of protocols. This composition theorem will be the main source of our privacy proofs.

We now review some standard tools, which we shall use. These tools require $O(1)$ rounds of communication. We measure their running time in terms of bit operations in $m = \lceil \log_2 P \rceil$ (the modulus length) and n (the number of servers). Below, we use $\mathcal{B}$ as a shorthand for $O(nm^2 + mn^2 \log n)$.

Sharing a secret. To compute a Shamir sharing of $x \in \mathbb{Z}_P$ over $\mathbb{Z}_P$, player P_j chooses random coefficients $\alpha_k \in \mathbb{Z}_P$ for $k = 1, \ldots, \tau$. He then sends $[x]_i^P = x + \sum_{k=1}^{\tau} \alpha_k \cdot i^k \bmod P$ to player P_i. We denote this protocol by $\texttt{RANDSS}(x)$; it takes $O(n^2 m \log n)$ bit operations.

Basic operations. Addition and multiplication of a constant and a Shamir share can be done locally. Hence, $[x]_j^P + c \bmod P$ is a polynomial share of $x + c \bmod P$ and $c \cdot [x]_j^P \bmod P$ is a share of $xc \bmod P$. These operations take $O(m)$ and $O(m^2)$ bit operations, respectively. Similarly, we can compute $[x]_j^P + [y]_j^P \bmod P$, which is a share of $x + y \bmod P$. Addition requires $O(m)$ bit operations.

Multiplication. Product of polynomially many shared secrets $x_1, \ldots, x_s \in \mathbb{Z}_P^*$ can be computed in constant rounds [BIB89, Kil05]. We denote this protocol by $\mathtt{MUL}([x_1]_j^P, \ldots, [x_s]_j^P)$; it uses $O(s\mathcal{B})$ bit operations.

Conversion between bit shares. Given Shamir shares of a single bit $b \in \{0,1\}$ in $\mathbb{Z}_P$, we may need to obtain its shares in $\mathbb{Z}_Q$. We can do this as follows. First, each server P_j locally computes $[b']_j^P \leftarrow -2 \cdot [b]_j^P + 1 \pmod{P}$ to convert the bit from a 0/1 to a $1/-1$ encoding. Next, P_j chooses a random $b_j \in \{1, -1\}$ and shares it among servers in both $\mathbb{Z}_P$ and $\mathbb{Z}_Q$. He computes $[b'']_j^P \leftarrow \mathtt{MUL}([b']_j^P, [b_1]_j^P, \ldots, [b_n]_j^P)$ and reveals it for all servers to reconstruct b''. Finally, P_j multiplies $b'' \pmod{Q}$ by its share of $\mathtt{MUL}([b_1]_j^Q, \ldots, [b_n]_j^Q)$ and converts the result to a 0/1 encoding. The protocol requires $O(1)$ rounds and $O(n\mathcal{B})$ bit operations.

Bit representation. Let $x \in \mathbb{Z}_P$ be a shared secret (written $x_m \ldots x_1$ in binary). In some situations, we will need to obtain Shamir shares of the bits of x. For this, we will use a protocol by Damgård *et al.* [DFK$^+$06], denoted $([x_1]_j^P, \ldots, [x_m]_j^P) \leftarrow \mathtt{BITS}([x]_j^P)$, which uses $O((m \log m)\mathcal{B})$ bit operations.

Occasionally, we will need to compute shares of a least significant bits of $x \in \mathbb{Z}_P$ in $\mathbb{Z}_Q$ (rather than in $\mathbb{Z}_P$). We will first run the $\mathtt{BITS}([x]_j^P)$ protocol and then convert each bit share from $\mathbb{Z}_P$ to $\mathbb{Z}_Q$. We denote this protocol by $([x_1]_j^Q, \ldots, [x_a]_j^Q) \leftarrow \mathtt{BITS}([x]_j^P, a, \mathbb{Z}_Q)$. It requires $O(1)$ rounds and $O((an + m \log m)\mathcal{B})$ bit operations.

Given bit-by-bit shares of $x \in \mathbb{Z}_P$, denoted $[x_1]_j^P, \ldots, [x_m]_j^P$, we can easily obtain shares of x by locally computing $[x]_j^P \leftarrow \sum_{i=1}^m 2^{i-1} \cdot [x_i]_j^P \bmod P$. This takes $O(m^3)$ bit operations.

Inversion. Let $x \in \mathbb{Z}_P$ be a shared secret. A protocol due to Bar-Ilan and Beaver [BIB89], denoted by $\mathtt{INV}([x]_j^P)$, allows us to compute the shares of $x^{-1} \bmod P$.

It takes an expected number of $O(\mathcal{B})$ bit operations.

Generating a random number. Occasionally, servers may need to jointly generate a shared random number. A simple protocol, denoted $\texttt{JRP}(\mathbb{Z}_P)$, accomplishes this in $O(mn^2 \log n)$ bit operations [ACS02]. There also exists a protocol $\texttt{JRPZ}(\mathbb{Z}_P)$ to jointly compute a sharing of zero modulo P in $O(mn^2 \log n)$ bit operations.

Exponentiation Some of our protocols require computing the shares of $x^y \bmod P$ when: (i) the exponent $y \in \mathbb{Z}_Q$ is shared, but the base is fixed; (ii) the base $x \in \mathbb{Z}_P$ is shared and the exponent is fixed, or (iii) both the base and the exponent are shared. We denote protocols for the above scenarios $\texttt{EXP}_1(x, [y]_j^Q)$, $\texttt{EXP}_2([x]_j^P, y)$, and $\texttt{EXP}([x]_j^P, [y]_j^Q)$. They run in $O(1)$ rounds and require, respectively, $O((mn + m\log m)\mathcal{B}), O(m^3 + n\mathcal{B})$, and $O(m^4 + (mn + m\log m)\mathcal{B})$. bit operations per player. They appear in Section 5.6.

5.3 Distributed Pseudo-Random Functions

In this section, we describe two distributed PRF constructions, where both the secret key and the input are shared. This will ensure that unscrupulous servers do not learn the results of intermediate Luby-Rackoff computations. In Section 5.3.1, we show how to do this for the PRF by Naor and Reingold [NR97]. Then in Section 5.3.2, we describe how to do this for our PRF from Chapter 4.

Let the input size $l : \mathbb{N} \mapsto \mathbb{N}$ be a function computable in poly(k) time. Sometimes, for simplicity, we will write l for $l(k)$. The initial input for all servers is a triple (P, Q, g), where P, Q are large primes such that $P = 2Q + 1$ and $P \equiv 3 \bmod 4$. Here g is a generator of quadratic residues subgroup G_Q of $\mathbb{Z}_P^*$. The group $\mathbb{Z}_P^*$ must be sufficiently large, *i.e.* $P \gg 2^k$. Such a triple can be publicly chosen without a trusted party by executing Bach's algorithm [Bac85].

Both centralized PRFs take as input an l-bit message x, the secret key SK and output a random group element in G_Q. In our distributed PRFs, each server P_j receives a share of the secret key SK and l shares of bits of x.

5.3.1 Naor-Reingold PRF

We consider a well-known PRF construction by Naor and Reingold [NR97]. Its secret key $SK = (a_0, a_1, \dots, a_l)$ consists of $l+1$ random exponents in $\mathbb{Z}_Q$. Given an l-bit input $x = x_1 \dots x_l$, the PRF $F^{NR}_{SK} : \{0,1\}^l \mapsto G_Q$ is defined as

$$F^{NR}_{SK}(x) = (g^{a_0})^{\prod_{i\,:\,x_i=1} a_i}.$$

This PRF was shown to be secure for polynomially sized inputs, $l(k) = \text{poly}(k)$, under the decisional Diffie-Hellman (DDH) assumption: "given (g, g^x, g^y) and $R \in G_Q$, it is hard to determine if $R = g^{xy}$ or not."

We can compute the PRF value recursively. Set $h_0 = g^{a_0}$. Then, for all $i = 1, \dots, l$,

$$h_i = \begin{cases} h_{i-1}^{a_i} & \text{if } x_i = 1, \\ h_{i-1} & \text{otherwise.} \end{cases} \tag{5.1}$$

It is easily seen that the PRF value must be equal to h_l. This form is convenient for distributed computation when both the input x and the secret exponents a_i are shared. One problem here is that we need to implement an if-condition on secret input x. We can use a simple trick and rewrite Equation (5.1) as

$$h_i = h_{i-1}(1 - x_i) + h_{i-1}^{a_i} x_i \text{ for } x_i \in \{0,1\}. \tag{5.2}$$

Computing the PRF value distributively amounts to several rounds of distributed multiplication and exponentiation:

Algorithm 1: A protocol $\texttt{PRF-NR}(([a_0]_j^Q, \ldots, [a_l]_j^Q), ([x_1]_j^P, \ldots, [x_l]_j^P))$ for distributed computation of $F_{SK}^{NR}(x)$.

```
1  $[0]_j^P \leftarrow \texttt{JRPZ}(\mathbb{Z}_P)$            ▷Servers jointly generate a sharing of 0 mod $P$
2  $[h_0]_j^P \leftarrow [0]_j^P + g \bmod P$        ▷And compute a share of generator $g$.
3  for $i \leftarrow 1$ to $l$                            ▷For all input bits $i$,
4  do
5      $[r]_j^P \leftarrow \texttt{MUL}([h_{i-1}]_j^P, 1 - [x_i]_j^P)$
6      $[s]_j^P \leftarrow \texttt{EXP}([h_{i-1}]_j^P, [a_i]_j^Q)$
7      $[t]_j^P \leftarrow \texttt{MUL}([s]_j^P, [x_i]_j^P)$     ▷we compute shares of Equation (5.2)
8      $[h_i]_j^P \leftarrow [r]_j^P + [t]_j^P$
9  end
10 return $[h_l]_j^P$                              ▷Return a share of the PRF value.
```

Proving security of this protocol is straightforward given the security of its sub-protocols by the composition theorem. The size of the secret key is proportional to the length of the input. What is worse, this protocol requires $O(l)$ rounds of communication. The running time is dominated by l calls to exponentiation protocol in line 6, yielding $O(m^4 l + (mn + m \log m)\mathcal{B}l)$ bit operations per player.

5.3.2 Our PRF

Recall that our pseudo-random function $F_{SK}^{DY} : \{0,1\}^{l(k)} \mapsto G_Q$ from Chapter 4 is as follows: given an l-bit input x (which can also be thought of as an element in $\mathbb{Z}_Q$) and the secret key $SK \in \mathbb{Z}_Q$, the function value is $F_{SK}^{DY}(x) = g^{1/(x+SK)}$ [DY05]. Our proof of security relied on an unorthodox q**-decisional Diffie-Hellman inversion** (q-DDHI) assumption: "given the tuple $\left(g, g^x, \ldots, g^{(x^q)}\right)$ and $R \in G_Q$ as input, it is hard to decide whether $R = g^{1/x}$ or not." Specifically, we showed:

Theorem 20 (Dodis-Yampolskiy) *Suppose an attacker who runs for $s(k)$ steps cannot break the $2^{l(k)}$-DDHI assumption in group G_Q with advantage $\epsilon(k)$. Then* *no algorithm running in less than $s'(k) = s(k)/(2^{a(k)} \cdot poly(k))$ steps can distinguish $F^{DY}_{SK}(\cdot)$ from a random function with advantage $\epsilon'(k) = \epsilon(k) \cdot 2^{l(k)}$.*

Because the security reduction is rather loose, we can construct PRFs only with small superlogarithmic input $l(k) = \omega(\log k)$. Unfortunately, "as is" this PRF is unsuitable for use in the Feistel transformation. A Feistel transformation uses length-preserving PRFs which map $l(k) = \text{poly}(k)$ input bits to $l(k)$ pseudo-random bits. In theory, small inputs are not a problem. We can either (1) shrink the inputs using a collision-resistant hash function [Dam87] or (2) utilize the generic tree construction [GGM86] to extend the input range. However, when we need to distribute the computation of this PRF between different servers, neither of these options becomes acceptable. As of today, we do not know how to efficiently distribute collision-resistant hash functions. And if we decide to utilize the generic tree construction, then we might as well use the Naor-Reingold PRF from the start.

Instead, we assume subexponential hardness of the q-DDHI assumption in G_Q; that is, we suppose that there is no way to break the q-DDHI assumption except by computing the discrete logarithm of g^x in $\mathbb{Z}^*_P$. The fastest algorithm for computing discrete logarithms modulo P runs in time roughly $\exp((1 + o(1)) \cdot \sqrt{\log P}\sqrt{\log \log P})$ [COS86]. It seems reasonable to assume that no algorithm running in time less than $s(k) = 2^{k^{\epsilon_2}}$ (for some small $\epsilon_2 \approx \frac{1}{3}$) can break the q-DDHI assumption. Formally:

Definition 21 (strong DDHI assumption) *We say that the strong DDHI assumption holds in G_Q if there exist $0 < \epsilon_1 < \epsilon_2$ such that for all probabilistic families* *of Turing machines* $\{A_k\}_{k\in\mathbb{N}}$ *with running time* $O(2^{k^{\epsilon_2}})$ *and* $q \leq 2^{k^{\epsilon_1}}$*, we have:*

$$\Big| \Pr_x \left[\mathcal{A}_k(g, g^x, \ldots, g^{(x^q)}, R) = 1 \mid R \leftarrow g^{1/x}\right] - \Pr_x \left[\mathcal{A}_k(g, g^x, \ldots, g^{(x^q)}, R) = 1 \mid R \xleftarrow{\$} G_Q\right] \Big| \leq poly(k)/2^{k^{\epsilon_2}},$$

where the probability is taken over the coin tosses of $\mathcal{A}_k$ *and the random choice of* $x \in \mathbb{Z}_Q^*$ *and* $R \in G_Q$*.*

By Theorem 20, the strong DDHI assumption immediately allows us to support inputs of size k^{ϵ_1} for small $\epsilon_1 > 0$. We show in Section 5.7 that this assumption is reasonable in the generic group model [Sho97].

What we need is a shrinking hash function, which maps long $l(k) = k$ bit inputs to smaller $a(k) = k^{\epsilon_1}$ bit inputs, which can be used as an input to $F_{SK}^{DY}(\cdot)$. A typical tool used for this purpose is a family of δ-universal hash functions $\mathcal{H} = \{h_i : \{0,1\}^l \mapsto \{0,1\}^a\}_{i\in\mathbb{Z}_Q^*}$.[2] The simplest such construct is

$$h_i(x) = (ix \bmod Q) \bmod 2^a,$$

where the collision probability $\delta = 1/2^a = 1/2^{k^{\epsilon_1}}$ is the best we can hope for.

We can thus define a new function $F' : \{0,1\}^l \mapsto G_Q$ as

$$F'_{SK,i}(x) = F_{SK}^{DY}(h_i(x)),$$

which is easily seen to be a secure PRF for polynomially sized inputs using a standard hybrid argument.

[2]We say that a hash family is δ-universal if, for all distinct inputs $x, x' \in \{0,1\}^l$, we have $\Pr_i[h_i(x) = h_i(x')] \leq \delta$.

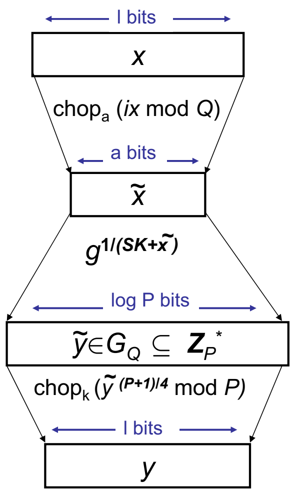

Figure 5.1: Transformation of $F^{DY}_{SK}(\cdot)$ into a length-preserving PRF.

This new PRF can be used in the Feistel transformation. We now describe how to distribute its computation:

Algorithm 2: A protocol $\texttt{PRF-DY}([i]^Q_j, [SK]^Q_j, [x_1]^Q_j, \ldots, [x_l]^Q_j)$

```
1 $[x]^Q_j \leftarrow \sum_{i=0}^{l-1} 2^i \cdot [x_{i+1}]^Q_j \bmod Q$            $\triangleright$Encode input $x$ as an element in $\mathbb{Z}^*_Q$.
2 $[r]^Q_j \leftarrow \texttt{MUL}([i]^Q_j, [x]^Q_j)$                              $\triangleright$Then hash it to $ix \bmod Q$.
3 $a \leftarrow \lfloor l^{1/3} \rfloor$                                            $\triangleright$Shrinking factor $a = l^{1/3}$.
4 $([r_1]^Q_j, \ldots, [r_a]^Q_j) \leftarrow \texttt{BITS}([r]^Q_j, a, \mathbb{Z}_Q)$   $\triangleright$Chop all but $a$ least significant bits.
5 $[\tilde{x}]^Q_j \leftarrow \sum_{i=0}^{a-1} 2^i \cdot [r_{i+1}]^Q_j \bmod Q$
6 $[s]^Q_j \leftarrow [\tilde{x}]^Q_j + [SK]^Q_j \bmod Q$                          $\triangleright$A share of $(\tilde{x} + SK)$.
7 $[t]^Q_j \leftarrow \texttt{INV}([s]^Q_j)$                                        $\triangleright$Invert the share into $1/(\tilde{x} + SK)$.
8 $[y]^P_j \leftarrow \texttt{EXP}_1(g, [t]^Q_j)$                                   $\triangleright$Exponentiate to get shares of $g^{1/(\tilde{x}+SK)}$.
9 **return** $[y]^p_j$
```

The security of the protocol again follows by composition theorem from security of its subcomponents. Unlike Algorithm 1, this algorithm uses $O(1)$ rounds of communication. However, it relies on a rather strong complexity assumption. Line 8 dominates the running time. It requires $O((mn + m \log m)\mathcal{B})$ bit operations, which is (more than) l times cheaper than the Naor-Reingold distributed protocol.

5.4 Distributed Pseudo-Random Permutations

We now show how to construct a **threshold pseudo-random permutation** by distributing the Luby-Rackoff construction. In principle, the Luby-Rackoff construction can be used with any PRF. However, we will use it with our PRF from Chapter 4, which allows us to evaluate the threshold PRP with only $O(1)$ communication rounds.

We begin by reviewing some formal definitions in Section 5.4.1. In Section 5.4.2, we show how to distribute a single Feistel permutation. In Section 5.4.3, we put all of the pieces together and explain how to distribute the entire Feistel cascade. Finally, in Section 5.4.4, we analyze our protocol's security and sketch how to make it proactive.

5.4.1 Definitions

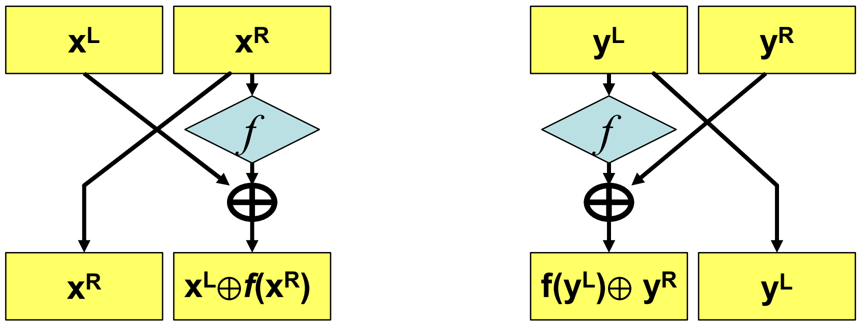

Figure 5.2: The Feistel transform and inverse Feistel transform.

Definition 22 (Feistel transformation) *Let $F : \{0,1\}^l \mapsto \{0,1\}^l$ be an l-bit mapping. We denote by $\bar{F}$ the permutation on $\{0,1\}^{2l}$ defined as $\bar{F}(x) = (x^R, x^L \oplus F(x^R))$, where $x = (x^L, x^R)$. Note that $\bar{F}$ is a permutation even if F is not. Its inverse is given by $\bar{F}^{-1}(y^L, y^R) = (f(y^L) \oplus y^R, y^L)$.*

Definition 23 (Feistel network) *Let $F_1, \ldots, F_k : \{0,1\}^l \mapsto \{0,1\}^l$ be l-bit mappings. Then a k-round Feistel network is a composition*

$$\Psi(F_1, \ldots, F_k) = \bar{F}_1 \circ \bar{F}_2 \cdots \bar{F}_k$$

Theorem 24 (Luby-Rackoff) *The permutation on $\{0,1\}^{2l}$ defined by $\Psi(F_1, F_2, F_3, F_4)$ cannot be distinguished from a random permutation by a PPT adversary. Here, F_i are independently keyed pseudo-random functions.*

5.4.2 Distributed Feistel Transformation

In Section 5.3.2, we defined a PRF acting on $l(k) = \text{poly}(k)$ bit inputs by $F'_{i,SK}(x) = F^{DY}_{SK}(h_i(x))$. We also gave an $O(1)$ round protocol `PRF-DY` that computes shares of a PRF value $g^{1/(h_i(x)+SK)}$ from shares of input's bits and secret key. We now show how to distribute the Feistel transformation $\overline{F'_{i,SK}}$, which maps (x^L, x^R) to $(x^R, x^L \oplus F'_{i,SK}(x^R))$. The inverse Feistel transformation can be computed in a similar manner.

Our PRF protocol outputs shares of a random group element in G_Q. Meanwhile, we need a sharing of a random l-bit string to use in the Feistel transformation. We use a deterministic extractor, which does not lose any entropy, to extract l bits of randomness. In the centralized setting, given PRF output $\tilde{y} \in G_Q$, we can simply compute its square root by letting $y = (\tilde{y}^{(P+1)/4} \bmod P) \bmod 2^l$ (see also Figure 5.1). To distribute the extractor, we use a distributed exponentiation protocol followed by a conversion into bit shares. Notice that if we have shares of bits $x_i, y_i \in \{0,1\}$, denoted by $[x_i]^Q_j$ and $[y_i]^Q_j$, we can compute a share of their exclusive-OR as $[z_i]^Q_j \leftarrow [x_i]^Q_j + [y_i]^Q_j$ and $[z_i]^Q_j \leftarrow \texttt{MUL}([z_i]^Q_j, 2 - [z_i]^Q_j \bmod Q)$.

Therefore, given bit shares of a $2l$-bit input (x^L, x^R), we can readily compute bit shares of a Feistel transformation:

Algorithm 3: One round of Feistel transformation $\texttt{FEISTEL}([i]^Q_j, [SK]^Q_j, [x_1]^Q_j, \ldots, [x_{2l}]^Q_j)$.

```
1 $[\tilde{y}]^P_j \leftarrow \texttt{PRF-DY}([i]^Q_j, [SK]^Q_j, [x_{l+1}]^Q_j, \ldots, [x_{2l}]^Q_j)$  ▷Compute the PRF value on $x^R$.
2 $[y]^P_j \leftarrow \texttt{EXP}_2([\tilde{y}]^P_j, (P+1)/4)$
3 $([y_1]^Q_j, \ldots, [y_l]^Q_j) \leftarrow \texttt{BITS}([y]^P_j, l, \mathbb{Z}_Q)$  ▷Compute bit shares of
    $(y^{(p+1)/4} \bmod p) \bmod 2^l$.
4 for $i \leftarrow 1$ to $l$ (in parallel)  ▷For all bits $i$
5 do
6     $[z_i]^Q_j \leftarrow [x_i]^Q_j + [y_i]^Q_j \bmod Q$
7     $[z_i]^Q_j \leftarrow \texttt{MUL}([z_i]^Q_j, 2 - [z_i]^Q_j \bmod Q)$  ▷We compute a share of $x_i \oplus y_i$.
8 end
9 return $([x_{l+1}]^Q_j, \ldots, [x_{2l}]^Q_j, [z_1]^Q_j, \ldots, [z_l]^Q_j)$  ▷Return shares of
    $(x^R, x^L \oplus F^{DY}_{SK}(x^R))$.
```

Security follows from composition theorem and security of its subprotocols. The protocol requires $O(1)$ rounds of communication between servers, because the for-loop is computed in parallel, and all other primitives take $O(1)$ rounds. The bit complexity is dominated by a call to the `PRF-DY` protocol in line 1, yielding $O((mn + m\log m)\mathcal{B})$ bit operations per player.

5.4.3 Distributed Luby-Rackoff Construction

Once we have a distributed protocol for the Feistel transformation, it is easy to distribute the Luby-Rackoff construction of PRP $g_s(x) = \Psi(F_1, F_2, F_3, F_4)(x)$. Initially, the n servers own shares of four independently chosen secret keys for the PRFs. These keys may either be jointly generated by servers or distributed to servers by a trusted party. An untrusted user, who wants to evaluate the PRP on input $x = (x^L, x^R)$, broadcasts x to the servers.[3] The servers convert x into bit shares and then run the distributed Feistel transformation for four rounds. We thus get:

[3]Alternatively, the user can split x into bit shares himself.

Algorithm 4: LUBY-RACKOFF$(([i_1]_j^Q, [SK_1]_j^Q), \ldots, ([i_4]_j^Q, [SK_4]_j^Q), x)$

```
1  $[0]_j^Q \leftarrow$ JRPZ$(\mathbb{Z}_Q)$                                  ▷Shares of zero.
2  for $i \leftarrow 1$ to $2l$ (in parallel)        ▷Locally compute shares of input's bits.
3  do
4      $[y_i]_j^Q \leftarrow [0]_j^Q + x_i \bmod Q$
5  end
6  for $rnd \leftarrow 1$ to $4$                  ▷Run the Feistel transformation for four rounds.
7  do
8      $([y_1]_j^Q, \ldots, [y_{2l}]_j^Q) \leftarrow$ FEISTEL$([i_{rnd}]_j^Q, [SK_{rnd}]_j^Q, [y_1]_j^Q, \ldots, [y_{2l}]_j^Q)$
9  end
10 return $([y_1]_j^Q, \ldots, [y_{2l}]_j^Q)$
```

The round complexity is $O(1)$. Bit complexity is dominated by four calls to the Feistel protocol, which take $O((mn + m \log m)\mathcal{B})$ bit operations per player.

Similarly, we can distribute the inverse permutation $g_s^{-1}(\cdot)$ by replacing calls to Feistel transforms with calls to inverse Feistel transforms. We denote the resulting protocol by LUBY-RACKOFF^{-1}. The round and bit complexity remain the same.

5.4.4 Security

In the stand-alone case, the security of a PRP $g_s(\cdot) : \{0,1\}^{2l} \mapsto \{0,1\}^{2l}$ is formalized via a game between an attacker and an oracle. The attacker can query the oracle for $g_s(\cdot)$ and $g_s^{-1}(\cdot)$ on messages of his choice. Roughly, the PRP is deemed secure if no attacker can tell apart $g_s(x^*)$ from random for any message x^*, which was not asked as a query.

In the distributed setting, the attacker also gets transcripts of semi-honest servers. The **security property of threshold PRP** states that these transcripts do not help the attacker in any way. Formally, for any PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the security of threshold PRP by corrupting servers $P_{i_1}, \ldots, P_{i_\tau}$, there exists a PPT $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the security of the original PRP. The attacker $\mathcal{A}$ learns key shares of corrupted servers. Then $\mathcal{A}_1$ runs in the first stage where it can interact with any honest servers on inputs of his choice. Attacker can ask servers either **encryption queries** where he learns shares of $g_s(x)$ or **decryption queries** for $g_s^{-1}(y)$. At the end of the phase, $\mathcal{A}_1$ outputs state information for $\mathcal{A}_2$ and a challenge input x^*, whose PRP value was not asked as a query. In the second stage, a random coin $b \in \{0,1\}$ is tossed. $\mathcal{A}_2$ receives a challenge Γ_b, which is either $\Gamma_0 \leftarrow g_s(x^*)$ or $\Gamma_1 \overset{\$}{\leftarrow} \{0,1\}^{2l}$. We let $\mathcal{A}_2$ interact with honest servers, but prohibit it from asking encryption queries on x^* or decryption queries on Γ_b. Finally, $\mathcal{A}_2$ outputs a guess b'. We say that $\mathcal{A}$ breaks the scheme if $\Pr[b = b'] > 1/2 + negl(k)$.

Theorem 25 `LUBY-RACKOFF` *protocol is an* $\lfloor \frac{n-1}{2} \rfloor$*-secure threshold pseudo-random permutation in the static, honest-but-curious setting.*

Proof: In the honest-but-curious setting, `LUBY-RACKOFF` protocol correctly computes a permutation $g_s(x) = \Psi(F_1, F_2, F_3, F_4)(x)$ for some secret key $s = ((i_1, SK_1), \ldots, (i_4, SK_4))$. We thus concentrate on the pseudorandomness property.

For sake of contradiction, suppose there exists adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the security of `LUBY-RACKOFF`. Since $\mathcal{A}$ is static, we assume it corrupts the maximum allowed threshold of servers before the protocol starts[4]. By symmetry, we can assume corrupt servers P_j have indices $Bad = \{1, \ldots, \tau\}$. Bad servers learn their shares of secret key s. They also observe the protocol's input x, output $y = g_s(x)$, shares of output's bits $y_1, \ldots, y_{2l}$ of both good and bad servers, and all messages Ξ exchanged during the protocol. The adversarial view is thus a random variable

$$\mathrm{VIEW}_{\mathrm{LUBY},\mathcal{A}} = \left\langle ([i_1]_k^Q, [SK_1]_k^Q), \ldots, ([i_4]_k^Q, [SK_4]_k^Q),\ x, y, [y_1]_j^Q, \ldots, [y_{2l}]_j^Q,\ \Xi \right\rangle_{1 \le j \le n,\ k \in Bad}.$$

We construct a simulator $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the security of a PRP $g_s(\cdot)$. It

[4]If not, we can arbitrarily fix some of the honest servers to be corrupt.

will run $\mathcal{A}$ in a virtual distributed environment and imitate $\mathcal{A}$'s replies to distinguish $g_s(\cdot)$ from a random permutation, thereby violating Theorem 24.

Setup: Algorithm $\mathcal{B}$ generates random shares of keys for corrupt servers. For $j \in$ *Bad*, it picks $([i_1]_j^Q, [SK_1]_j^Q), \ldots, ([i_4]_j^Q, [SK_4]_j^Q) \xleftarrow{\$} \mathbb{Z}_Q^* \times \mathbb{Z}_Q^*$ and gives them to $\mathcal{A}$.

Responding to queries: When $\mathcal{A}$ initiates an honest server P_j ($j \notin$ *Bad*) on input x, $\mathcal{B}$ in turn asks his oracle for $y = g_s(x)$. It generates random output shares $[z_1]_j^Q, \ldots, [z_{2l}]_j^Q \xleftarrow{\$} \mathbb{Z}_Q^*$ for $j \in$ *Bad*. Then, $\mathcal{B}$ augments the set of shares of corrupted servers into a full and random sharing of y's bits. For each bit $y_i \in \{0,1\}$ ($1 \leq i \leq 2l$), $\mathcal{B}$ picks a random polynomial $\alpha_i(x) \in \mathbb{Z}_Q[X]$ satisfying $\alpha_i(j) = [z_i]_j^Q$ and $\alpha_i(0) = y_i$. The adversary $\mathcal{A}$ receives randomized output shares $(\alpha_1(j), \ldots, \alpha_{2l}(j))$ for all servers P_j ($1 \leq j \leq n$). In the semi-honest setting, we can simulate the transcript of each subprotocol used by `LUBY-RACKOFF` given its input and output values. We can thus use these protocols as black-boxes and simulate messages Ξ in $\textsc{View}_{\textsc{Luby},\mathcal{A}}$. These values provide a perfect simulation of the coalition's view. Decryption queries are handled just like encryption queries except $\mathcal{B}$ queries another oracle $g_s^{-1}(\cdot)$.

Challenge: Eventually, attacker $\mathcal{A}$ outputs a message x^* on which it wants to be challenged. It claims to be able to distinguish output of `LUBY-RACKOFF`(x^*) from a random $2l$-bit string. $\mathcal{B}$ sends the same challenge x^* to the trusted party and gets back Γ, which is either $g_s(x^*)$ or a random string. Finally, $\mathcal{B}$ gives Γ to $\mathcal{A}$.

Guess: Attacker $\mathcal{A}$ continues to issue queries for messages other than x^*. Simulator $\mathcal{B}$ responds to queries as before. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$, which $\mathcal{B}$ also returns as its guess.

■

An adversary who controls less than $\tau = \lfloor (n-1)/2 \rfloor$ servers cannot break the privacy of our protocol. The protocol can easily be amended to achieve **proactive security** [HJKY95] and withstand the compromise of even all servers as long as at most τ servers are corrupted during each time period. The basic idea is to have servers periodically refresh their shares of the input and the secret keys. To be exact, each server P_j will from time to time execute the `JRPZ` protocol to generate a random share of zero, called $[0]_j^Q$. It will then update its input share to $[x]_j^Q \leftarrow [x]_j^Q + [0]_j^Q$ and its secret keys' shares to $[SK_i]_j^Q \leftarrow [SK_i]_j^Q + [0]_j^Q$.

5.5 Applications of the Distributed Block Cipher

In the previous section, we have constructed a **threshold pseudorandom permutation**. Whenever a PRP is used as part of the construction, we can plug in our protocol instead.

Let $g_s : \{0,1\}^{2l} \mapsto \{0,1\}^{2l}$ be a $2l$-bit pseudo-random permutation obtained from the Luby-Rackoff construction $\Psi(F_1, F_2, F_3, F_4)$. The PRP's key s consists of four secret keys SK_i of pseudo-random functions F_i used in the construction. We denote by `LUBY-RACKOFF` our distributed protocol, which evaluates the $g_s(\cdot)$.

5.5.1 CCA-Secure Symmetric Encryption

A PRP is deterministic, so by itself it cannot be a secure encryption scheme [GM82]: The adversary can easily detect if the same message has been encrypted twice. Desai [Des00] described how a **CCA-secure symmetric encryption scheme** can be obtained from a strong PRP. The encryption $\mathcal{E}_s(m)$ of a $(2l-k)$-bit message m is defined as $\mathcal{E}_s(m) = g_s(m, r)$, where r is a k-bit randomly generated nonce. To decrypt, the user computes $\mathcal{D}_s(c) = g_s^{-1}(c)$ and extracts the message. To distribute the computation, the key s is split into shares among the n servers. Upon receiving a message m, the servers run the `JRP` protocol to generate shares of a secret random number r. They can extract shares of k bits, written $[r_1]_j^Q, \ldots, [r_k]_j^Q$, using the `BITS` protocol. Bit shares of m are easy to compute since m is public. Finally, the servers invoke `LUBY-RACKOFF` on $([m_1]_j^Q, \ldots, [m_{2l-k}]_j^Q, [r_1]_j^Q, \ldots, [r_k]_j^Q)$ to get shares of $g_s(m, r)$.

5.5.2 Authenticated Encryption

If we make the nonce r public and check during decryption that it matches the nonce in the ciphertext, then we get a distributed **authenticated encryption scheme** (AE). Encryption of m is given by $\mathcal{AE}_s(m) = (r, g_s(m, r))$. The decryption algorithm $\mathcal{AD}_s(r', c)$ computes $(r, m) = g_s^{-1}(c)$ and checks that $r = r'$ before returning m to the user. The message here is rather short: It is limited to $(2l - k)$ bits by the length of the PRP. For longer messages, we can use an amplification paradigm of Dodis and An [DA03]: We compute a **concealment** (b, h) of message m $(|b| \ll |m|)$, which is a specialized publicly known transformation. In fact, we can even implement distributed **remotely keyed authenticated encryption** (RKAE) [BFN98], where the servers do not need to perform any checks and just serve as PRP oracles for an untrusted user. The secret key s is split into shares among several computationally bounded **smartcards**, and an insecure, powerful **host** performs most of computations. The insecure host computes a concealment (b, h) of m and sends it to the smartcards, who run the `LUBY-RACKOFF` protocol, and return shares of $g_s(b)$.

5.5.3 Cipherblock Chaining Mode

We often need to encrypt messages that are longer than $2l$ bits. The message m is usually split into blocks $(m_1, \ldots, m_k)$ each of length $2l$. Then a PRP may be used in **cipher block chaining** (CBC) mode [MvOV97], which initializes c_0 with a random $2l$-bit string and sets $c_i = g_s(c_{i-1} \oplus m_i)$ for $i = 1, \ldots, k$. The encryption of m is defined to be $(c_0, c_1, \ldots, c_k)$. To decrypt, the user can compute $m_i = g_s^{-1}(c_i) \oplus c_{i-1}$. The servers own shares of secret key s. The untrusted user broadcasts message m to the servers. We must be careful to guard against the blockwise adaptive attacks [JMV02]; hence, we require the user to send an entire message m. The servers run the `JRP` protocol to generate a random shared number from which shares of a $2l$-bit c_0 are extracted. For $i = 1, \ldots, k$ rounds, the servers distributively XOR shares of c_{i-1} and m_i (as in Section 5.4.3), and then run the `LUBY-RACKOFF` protocol on the result.

5.5.4 Variable Input Block Ciphers

Existing block ciphers operate on blocks of fixed length (FIL). Often, one needs a block cipher that can operate on inputs of variable length (VIL). There exist centralized constructions for VIL ciphers, which use a FIL block cipher as a black box: most notably, CMC [HR03], EME* [Hal04] and an unbalanced Feistel network [PRS04]. Our threshold PRP enables us to distribute the computation of these modes. Besides basic arithmetic operations, these modes XOR the ciphertexts (to distribute, we would use `BITS`), evaluate the fixed-length block cipher (`LUBY-RACKOFF`), compute the universal hash function $h_{a,b}(x) = ax + b$ (`MUL`), and truncate the outputs (`BITS`).

5.6 Distributed Exponentiation

We describe how to distribute the computation of $x^y \bmod P$.

Earlier papers by Damgård *et al.* [DFK$^+$06] and Shoup *et al.* [ACS02] sketched how to implement distributed exponentiation for some of these scenarios. Unlike prior constructions, in our schemes the modulus P is publicly known rather than shared among the parties; this leads to simpler and more efficient protocols. We also allow the base and the exponent to be shared over different moduli P and Q.

5.6.1 Base x known, exponent y shared.

We can rewrite:

$$x^y = x^{\sum_{i=0}^{l-1} 2^i y_i} = \prod_{i=0}^{l-1} \left(y_i x^{2^i} + 1 - y_i \right) \bmod P.$$

Algorithm 5: Protocol $\texttt{EXP}_1(x, [y]_j^Q)$: Base x is publicly known and exponent y is given as shares.

```
1 $([y_1]_j^P, \ldots, [y_l]_j^P) \leftarrow \texttt{BITS}([y]_j^Q, \mathbb{Z}_P)$.        ▷Obtain shares of exponent's bits.
2 for $i \leftarrow 1$ to $l$ (in parallel) do
3     $[z_i]_j^P \leftarrow [y_i]_j^P \cdot x^{2^{i-1}} + (1 - [y_i]_j^P) \bmod P$ ▷square-and-multiply algorithm used
      to compute $x^{2^i}$
4 end
5 $[z]_j^P \leftarrow \texttt{MUL}([z_1]_j^P, \ldots, [z_l]_j^P)$                  ▷Compute the product using
6 return $[z]_j^P$                                      ▷unbounded fan-in multiplication protocol.
```

The cost of the protocol is dominated by the bit conversion protocol. Its bit complexity is $O((mn + m \log m)\mathcal{B})$ and its round complexity is $O(1)$.

5.6.2 Base x shared, exponent y publicly known.

Damgård *et al.* [DFK$^+$06] sketched how $x^y \bmod P$ can be computed in this scenario for any $x \neq 0$. Below, we give a detailed protocol implementing his idea.

Algorithm 6: $\texttt{EXP}_2([x]_j^P, y)$: Base x is given as shares, while exponent is publicly known.

```
2  $r_j \xleftarrow{\$} \mathbb{Z}_P^*$ ; $t_j \leftarrow r_j^y \bmod P$        $\triangleright$Choose a blinding factor $r_j$ and $r_j^y$.
3  RANDSS($r_j$) ; RANDSS($t_j$)        $\triangleright$Share $r_j$ and $t_j$ between players.
4  upon receiving $(([r_1]_j^P, \ldots, [r_n]_j^P) \wedge ([t_1]_j^P, \ldots, [t_n]_j^P))$:
5      $[r]_j^P \leftarrow$ MUL($[r_1]_j^P, \ldots, [r_n]_j^P$)        $\triangleright$A jointly generated random $r$
6      $[t]_j^P \leftarrow$ MUL($[t_1]_j^P, \ldots, [t_n]_j^P$)        $\triangleright$and $t = r^y$.
7      $[z]_j^P \leftarrow$ MUL($[x]_j^P, [r]_j^P$)        $\triangleright$This is a blinded share of $xr$,
8      send($[z]_j^P$) to all players        $\triangleright$which we can reveal.

9  upon receiving $([z]_1^P, \ldots, [z]_n^P)$:
10     $z \leftarrow \sum_{j=1}^{\tau} \lambda_j [z]_j^P \bmod P$.        $\triangleright$Interpolate shares to get $z = xr$.
11     $w \leftarrow z^y \bmod P$
12     RANDSS($w$)        Share $w = (xr)^y$ between players.

13 return MUL($[w]_j^P$, INV($[t]_j^P$))        $\triangleright$A share of $z \cdot t^{-1} = (xr)^y \cdot (r^y)^{-1} = x^y \bmod P$.
```

This protocol is easily seen private as long as $x \neq 0$. It runs in $O(1)$ rounds. The running time is dominated by calls to multiplication protocol in lines 5-7 and by computation of Shamir secret sharing in lines 2-3. The bit complexity is thus $O(m^3 + n\mathcal{B})$.

5.6.3 Base x shared, exponent y shared.

Combining the above two protocols yields an exponentiation protocol where both the base and the exponent are shared. Essentially, the protocol is the same as $\texttt{EXP}_1$ except in line 5, we replace local multiplication with a call to distributed multiplication algorithm MUL. We also need to use $\texttt{EXP}_2$ to compute the shares of x^{2^i} for $i = 1, \ldots, l$. The final algorithm is as follows.

Algorithm 7: Protocol $\texttt{EXP}([x]_j^P, [y]_j^Q)$: Both the base and the exponent are given as shares.

```
1 $([y_1]_j^P, \ldots, [y_l]_j^P) \leftarrow \texttt{BITS}([y]_j^Q, \mathbb{Z}_P)$.  ▷Obtain shares of exponent's $l = \lfloor \log Q \rfloor$ bits.
2 for $i \leftarrow 1$ to $l$ (in parallel) do
3     $[s]_j^P \leftarrow \texttt{EXP}_2([x]_j^P, 2^{i-1} \bmod Q)$          ▷$s$ is a share of $x^{2^i}$.
4     $[z_i]_j^P \leftarrow \texttt{MUL}([y_i]_j^P, [s]_j^P) + (1 - [y_i]_j^P) \bmod P$    ▷A share of $y_i x^{2^i} + (1 - y_i)$.
5 end
6 $[z]_j^P \leftarrow \texttt{MUL}([z_1]_j^P, \ldots, [z_l]_j^P)$          ▷Compute the product using
7 return $[z]_j^P$                     ▷unbounded fan-in multiplication protocol.
```

This protocol also runs in $O(1)$ rounds. The protocol performs l invocations of $\texttt{EXP}_2$ and a single invocation of $\texttt{BITS}$ and $\texttt{MUL}$; hence, it requires $O(m^4 + (mn + m \log m)\mathcal{B})$ bit operations per player.

5.7 Generic Security of the q-DDHI Assumption

We now analyze the q-decisional Diffie-Hellman inversion (q-DDHI) assumption in the generic model.

Theorem 26 *Let $\mathcal{A}$ be a generic algorithm that breaks the q-DDHI assumption in G. If $\mathcal{A}$ makes at most q_G queries to the group oracle, then*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c} b \xleftarrow{\$} \{0,1\}; \\ \Gamma_b \leftarrow 1/x;\ \Gamma_{1-b} \xleftarrow{\$} \mathbb{Z}_P^*; \\ b' \leftarrow \mathcal{A}\left(\sigma(1), \sigma(x), \ldots, \sigma(x^q), \sigma(\Gamma_0), \sigma(\Gamma_1)\right) \end{array} \right] \leq \frac{1}{2} + O\left(\frac{q_G^2 q + q^3}{p}\right).$$

Proof: Our proof proceeds similar to the proof of generic security in Chapter 4. However, because we do not need to simulate queries to oracles computing the bilinear pairing, this proof is simpler. Instead of letting adversary $\mathcal{A}$ interact with the actual oracles, we play the following game:

We maintain a list of pairs $L = \{ (F_i, \sigma_i) : i = 0, \ldots, t-1 \}$, where σ_i are binary strings in $\{0,1\}^*$ and $F_i \in \mathbb{Z}_P[X, \Gamma_0, \Gamma_1]$ are the multivariate polynomials. At the start of the game, we set the polynomials to $(F_0 = 1, F_1 = X, \ldots, F_q = X^q)$ and also let $F_{q+1} = \Gamma_0, F_{q+2} = \Gamma_1$. The corresponding encodings $\sigma_0, \ldots, \sigma_{q+2}$ are set to random distinct strings in $\{0,1\}^*$. Initially, the list L has length $t = q+3$.

We begin the game by providing adversary $\mathcal{A}$ with encodings $(\sigma_0, \ldots, \sigma_{q+2})$. Queries go as follows:

Group action: Given a multiply/divide bit and two encodings σ_i, σ_j $(0 \leq i, j < t)$, we compute $F_t = F_i \pm F_j$ accordingly. If $F_t = F_k$ for some $k < t$, we set $\sigma_t = \sigma_k$; otherwise, we set σ_t to a random string distinct from all other encodings, and increment the list length t by 1.

Eventually, the adversary $\mathcal{A}$ halts with its guess $b' \in \{0,1\}$. At this point, we choose random $x, y \in \mathbb{Z}_P^*$ and consider $\Gamma_b = 1/x$ and $\Gamma_{1-b} = y$ for both choices of b. The adversary wins the game if the values we chose for indeterminates give rise to some non-trivial equality relationship between the simulated group elements. This happens if for some $i \neq j$ either of these hold:

$$\begin{cases} F_i(x, 1/x, y) - F_j(x, 1/x, y) = 0 \\ F_i(x, y, 1/x) - F_j(x, y, 1/x) = 0. \end{cases}$$

Notice that the adversary can manipulate the polynomials on the list L only through additions and subtractions. He cannot concoct a polynomial that contains $1/X$ from the monomials we gave him at the beginning. Therefore, the above inequalities can occur only as a result of a numerical cancellation. Because for all i, $\deg(F_i) \leq q$, by the Schwartz-Zippel Lemma [Sch80], the probability that $F_i - F_j$

vanishes is at most q/p. Hence, $\mathcal{A}$'s advantage is at most

$$\begin{aligned}\epsilon &\leq 2\binom{t}{2}\frac{q}{p} \\ &< 2(q_G + q + 3)^2\frac{q}{p} \\ &= O\left(\frac{q_G^2 q + q^3}{p}\right).\end{aligned}$$

It turns out that decisional Diffie-Hellman inversion assumption is asymptotically as hard as the stronger decisional bilinear Diffie-Hellman inversion assumption. ■

Chapter 6

Economic Model of Network Security*

The previous chapters introduced various security tools, which can be used to enhance security of distributed systems. However, often users may be reluctant to adopt new security technologies, preferring an insecure system to a relatively secure one. In this chapter, we show that there may exist sound economic reasons behind this seeming foolhardiness.

6.1 Overview

We propose a simple game for modeling containment of the spread of viruses on a graph of n nodes. Each machine's owner separately chooses whether or not to install anti-virus software at some known cost C without regard to the effect on other machines. If he installs the software, then his machine becomes immune to virus infection; otherwise, it remains susceptible and may suffer a loss L if a virus

*This chapter describes joint work with James Aspnes and Kevin Chang. Joan Feigenbaum provided many useful comments. Parts of it were reported in [ACY05].

that starts at a random initial point in the graph can reach it without being stopped by some intermediate node.

Overview of Our Results. We give a complete characterization of the Nash equilibria for the anti-virus software installation game. We show that finding either the most or least expensive equilibrium is **NP**-hard, but that some Nash equilibrium can be computed in $O(n^3)$ time and that any population of nodes will quickly converge to a Nash equilibrium by updating their strategies locally based on the other nodes' strategies. Unfortunately, the cost of any such Nash equilibrium may be badly suboptimal; the **price of anarchy** for this game is $\Theta(n)$ in the worst case. This shows that for many graphs and values of C and L, letting the users choose individually whether or not to inoculate their machines will give bad results.

We then consider the possibility of a centralized solution in which a dictator computes and enforces an optimal inoculation plan. We show that essentially the same argument that shows that extreme Nash equilibria are hard to find applies to the optimal solution as well. However, we show that the problem of finding an optimal inoculation plan reduces to a graph partition problem in which we are asked to partition the graph by removing m nodes; the quality of the partition is measured by the sum of the squares of the sizes of its components. We give a polynomial-time approximation algorithm (PTAS) that removes $O(\log^2 n)m$ nodes in order to achieve a partition with quality within $O(1)$ of the optimum.

Chapter Organization. In Section 6.2, we define a static non-cooperative game for installing anti-virus software on the network. We characterize Nash equilibria of this game in Section 6.3. Then in Section 6.4, we show that an optimum placement of anti-virus software is **NP**-hard to compute. The approximation algorithm for anti-virus placement is described in Section 6.5. Some previous work on related problems is discussed in Section 6.6.

6.2 Our Model

We represent network topology by an undirected graph $G = (V, E)$, where $V = \{0, 1, \ldots, n-1\}$ is a set of network hosts and $E \subseteq V \times V$ is a set of (bidirectional) communication links. Our basic model for installing anti-virus software is a one-round game with the following features:

Players. Our game has n players corresponding to nodes of the graph. Initially, all nodes are insecure and vulnerable to infection.

Strategies. We denote the **strategy** of i by a_i. Each node i has two possible actions: **do nothing** and risk being infected or **inoculate itself** by installing anti-virus software. Node i's strategy a_i is the probability that it inoculates itself.

Nodes' choices can be summarized in a **strategy profile** $\vec{a} \in [0,1]^n$. If a_i is 0 or 1, we say that node i adopts a **pure strategy**; otherwise, its strategy is **mixed**. We call nodes that install anti-virus software **secure** and denote the set of such nodes by $I_{\vec{a}}$. We associate an **attack graph** $G_{\vec{a}} = G - I_{\vec{a}}$ with $\vec{a}$. It is essentially the network graph with secure nodes and their edges removed (see also Figure 6.1). Note that both $I_{\vec{a}}$ and $G_{\vec{a}}$ are random variables unless all strategies are pure.

Attack model. After the nodes made their choices, the adversary picks some node uniformly at random as a starting point for infection. Infection then propagates through the network graph. Node i gets infected if it has no anti-virus software installed and if any of its neighbors become infected.

Individual costs. Suppose it costs C to install anti-virus software. If a node is infected, it suffers a loss equal to L. For simplicity, we assume that both C and L are known and are the same for all nodes. The cost of a mixed strategy $\vec{a} \in [0,1]^n$ to node i is

$$\text{cost}_i(\vec{a}) = a_i C + (1 - a_i) L \cdot p_i(\vec{a}).$$

Here $p_i(\vec{a})$ is the probability of node i being infected given the strategy profile $\vec{a}$, *conditioned on the event that node i does not install the anti-virus software.* It is equal to the probability that some vulnerable node reachable from i without passing through a secure node is the initial point of infection. For pure strategies, this is just k_i/n, where k_i is the size of the connected component containing i in the attack graph $G_{\vec{a}}$.

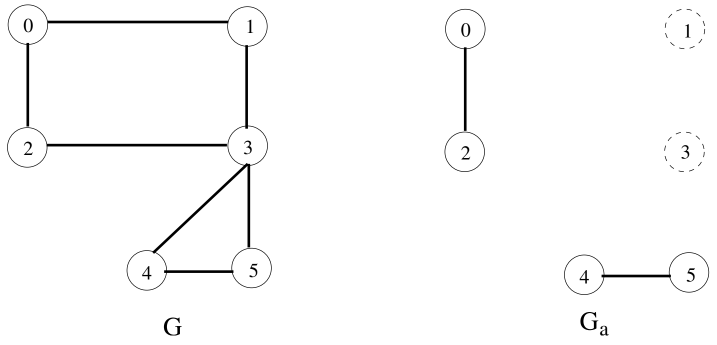

Figure 6.1: Sample graph G and its attack graph $G_{\vec{a}}$ for $\vec{a} = 010100$.

Social cost. The total social cost of a strategy profile is the sum of the individual costs. For pure strategies, there is a simple characterization of the total social cost in terms of the attack graph $G_{\vec{a}}$. Because each node in the same component of $G_{\vec{a}}$ has the same chance of infection, the k_i nodes in the i-th component between them face a loss of $k_i \cdot (Lk_i/n) = (L/n)k_i^2$. So the social cost is

$$\begin{aligned}
\text{cost}(\vec{a}) &= \sum_{j=0}^{n-1} \text{cost}_j(\vec{a}) \\
&= \sum_{j=0}^{n-1} a_j C + (1 - a_j) L \cdot p_j(\vec{a}) \\
&= C|I_{\vec{a}}| + \frac{L}{n} \sum_{i=1}^{l} k_i^2,
\end{aligned}$$

where $k_1, k_2, \ldots k_\ell$ are the sizes of the components in $G_{\vec{a}}$.

6.3 Nash Equilibria

We consider first the choices that the nodes will make in the absence of coordination, by examining the Nash equilibria of the game defined in Section 6.2. The assumption that the nodes will reach a Nash equilibrium is a very strong one, as it requires assuming that the nodes are aware of each other's choices to install or not and that the nodes can evaluate C (printed on the box for the anti-virus software) and L (which is more problematic). It also assumes that the nodes can compute a Nash equilibrium in a reasonable amount of time, which is not always possible for some games. However, we can show that Nash equilibria for our game are easily characterized in terms of the sizes of the components of the attack graph (Section 6.3.1), and that a population will converge to some Nash equilibrium quickly even though finding the best or worst *pure* equilibrium as measured by total cost is **NP**-hard (Section 6.3.2).

We can further imagine that some of the difficulties of limited information could be overcome by considering an iterated game where nodes pay C to rent the anti-virus software in each round and update their strategies based on observations of losses to viruses and the strategies of other nodes in previous rounds; though we do not analyze this multi-round game formally, a simplified version is implicit in our convergence result. We also show that the hardness of finding the worst-case equilibrium does not prevent obtaining further information about its behavior; for example, its total cost is nondecreasing as a function of the inoculation cost C (Section 6.3.3).

Unfortunately, selfish behavior proves to be highly undesirable, because the cost of a Nash equilibrium solution may be very far from the social optimum. In Section 6.3.4, we prove that while the **price of anarchy**, defined as the ratio of total cost between the worst Nash equilibrium and the social optimum never exceeds n, this bound is tight up to constant factors for some graphs and choices of C and L.

6.3.1 Characterization of Mixed and Pure Equilibria

A useful feature of the Nash equilibrium for our game is its simple characterization: there is always a component-size threshold $t = Cn/L$ such that each node will install the anti-virus software if it would otherwise end up in a component of vulnerable nodes with expected size greater than t, and will not install the software if it would otherwise end up in a component with expected size less than t. When the expected component size equals t, the node is indifferent between installing and not installing and may adopt a mixed strategy. The threshold arises in a natural way: it is the break-even point at which the cost C of installing the software equals the expected loss $L(t/n)$ of not installing.

We define $\vec{a}[i/x]$ to be the strategy vector that is identical to $\vec{a}$, except the i'th component a_i is replaced by x. Note that attack graph $G_{\vec{a}[i/0]}$ is the attack graph in which player i never installs the anti-virus software.

Theorem 27 *(Characterization of mixed equilibria): Suppose $S(i)$ is the expected size of the insecure component that contains node i of the attack graph $G_{\vec{a}[i/0]}$, (i.e.*

$S(i) = np_i(\vec{a})$).

Fix C, L. Let the threshold be $t = Cn/L$. A strategy profile $\vec{a}$ is a Nash equilibrium if and only if

(a) For all i such that $a_i = 1$, $S(i) \geq t$.

(b) For all i such that $a_i = 0$, $S(i) \leq t$.

(c) For all i such that $0 < a_i < 1$, $S(i) = t$.

Proof:

Suppose $\vec{a}$ is a Nash equilibrium and consider node i. The expected cost to node i is $a_i C + (1 - a_i)(L/n)S(i)$.

1. Suppose $a_i = 0$. Then node i has expected cost $(L/n)S(i)$. If $(L/n)S(i) > C$, then i will want find the situation $a_i = 1$ with cost C preferable. Thus, we must have $S(i) \leq CL/n = t$.

2. Suppose $a_i = 1$. Then node i has expected cost C. If $(L/n)S(i) < C$, then i would find the situation $a_i = 0$ with expected cost $(L/n)S(i) < C$ preferable. Thus, we must have $S(i) \geq CL/n = t$.

3. Suppose $0 < a_i < 1$. If $(L/n)S(i) > C$, then i will find the situation $a_i = 1$ preferable. If $(L/n)S(i) < C$, then i will find the situation $a_i = 0$ preferable. Thus, we must have $S(i) = CL/n = t$.

Thus, $\vec{a}$ satisfies condition (a), (b), and (c) above.

Conversely, suppose $\vec{a}$ satisfies conditions (a), (b) and (c) of the theorem. Consider node i.

1. Suppose $a_i = 0$. Then node i will have expected cost $(L/n)S(i) < C$, and thus will not want to switch to a different a_i that puts any weight on installing at cost C.

2. Suppose $a_i = 1$. Then node i will have cost C, and thus will not want to switch to a different a_i that puts any weight on being insecure at expected cost $(L/n)S(i) \geq C$.

3. Suppose $0 < a_i < 1$. Then node i will have expected cost $a_iC + (1 - a_i)(L/n)S(i) = C$. Switching to any other strategy will have the same expected cost.

Thus, $\vec{a}$ is a Nash equilibrium. ■

A special case of Theorem 27 is the following characterization for pure Nash equilibria. Because nodes in a pure Nash equilibrium do not make randomized choices, the attack graph is not a random object, but a determined graph. We have the same threshold conditions as before, but the removal of randomness simplifies the situation.

Corollary 28 *(Characterization of pure equilibria) Fix C, L. Let the threshold be $t = Cn/L$. A strategy profile $\vec{a}$ is a pure Nash equilibrium if and only if*

(a) Every component in attack graph $G_{\vec{a}}$ has size at most t.

(b) Inserting any secure node $j \in I_{\vec{a}}$ and its edges into $G_{\vec{a}}$ yields a component of size at least t.

For example, let $C = 0.5$ and $L = 1$, and consider the graph in Figure 6.1. The threshold for this graph is $t = Cn/L = 3$. Then Corollary 28 tells us that pure strategy $\vec{a} = 010100$ must be a Nash equilibrium for these C and L.

6.3.2 Computing Pure Nash Equilibria

Designing algorithms for finding mixed Nash equilibria or proving hardness results for finding optimized mixed equilibria would most likely involve estimating or otherwise manipulating the expected value of the sizes of components in the attack graph, which is at the very least a non-trivial problem. Furthermore, in the absence of central control, nodes attempting to calculate their best strategy based on a mixed strategy paradigm would possibly run into similar computational issues.

Thus, we turn our attention to the computation and hardness of pure Nash equilibria. Corollary 28 gives us a powerful tool with which to reason about pure Nash equilibria. We now show that computing the best or worst pure Nash equilibria is hard, but that finding some intermediate Nash equilibrium is easy. A consequence of this algorithm is that the existence of a pure Nash equilibrium is always guaranteed. (The existence of a mixed Nash equilibrium is a consequence of Nash's theorem.)

Theorem 29 *Both computing the pure Nash equilibrium with lowest cost and computing the pure Nash equilibrium with highest cost are **NP**-hard problems.*

Proof: We reduce VERTEX COVER to the decision problem "Does there exist a pure Nash equilibrium with cost less than c?" and we reduce INDEPENDENT DOMINATING SET to "Does there exist a pure Nash equilibrium with cost greater than c?"

Fix some graph $G = (V, E)$, and set $C/L = 1.5/n$ so that $t = Cn/L = 1.5$, where t is the component size threshold from Corollary 28. Then from Corollary 28, in any Nash equilibrium the components of the attack graph all have size at most 1, and any secure node is adjacent to some insecure node (as otherwise it could uninstall its software and be in a component of size at most 1). It follows that in a Nash equilibrium (a) every vulnerable node is either isolated or has all neighbors secure, and (b) every secure node has an insecure neighbor.

We now argue that G has a vertex cover of size k if and only if the inoculation game on G with the above settings of C and L has a Nash equilibrium with k or fewer secure nodes, or equivalently an equilibrium with social cost $Ck + (n-k)L/n$ or less, as each insecure node must be in a component of size 1 and contribute exactly L/n expected cost. Given a minimal vertex cover $V' \subseteq V$, observe that installing the software on all nodes in V' satisfies condition (a) because V' is a vertex cover, and (b) because V' is minimal. Conversely, if V' is the set of secure nodes in a Nash equilibrium, then V' is a vertex cover by condition (a). This concludes the proof that finding a minimum-cost Nash equilibrium is **NP**-hard.

For a maximum cost equilibrium, consider the set of *insecure* vertices. These consist of isolated nodes (which are already in components of size 1) and nodes that do not install the software because all their neighbors do. Given an independent dominating set $V' \subseteq V$, installing the software on all nodes *except* the nodes in V' satisfies condition (a) because V' is independent and (b) because V' is a dominating set. Conversely, the insecure nodes in any Nash equilibrium are independent by condition (a) and dominating by condition (b). This shows that G has an independent dominating set of size k if and only if it has a Nash equilibrium with no more than k insecure nodes, which occurs only if it has a Nash equilibrium with at least $n-k$ secure nodes or, equivalently, a cost of at least $C(n-k) + (L/n)(k)$. ■

Theorem 29 says that finding extreme pure equilibria is hard. But what if we just want to converge to some equilibrium, but we don't care which one? Suppose we implement the process of convergence implied by the Nash equilibrium: at each step, some participant, whose current strategy is suboptimal given the others' strategies, switches. This is an easy process to implement because each participant can detect if its strategy is suboptimal using the $t = Cn/L$ component size threshold from Corollary 28.[1] But does this process converge to a Nash equilibrium? If it does, how

[1]We must assume in this implementation either that the choice to install software or not is reversible, or that each player can observe the other players' intended actions and respond accordingly.

long does it take?

By choosing an appropriate potential function, we can show that this process does indeed converge to a Nash equilibrium quickly:

Theorem 30 *Starting from any pure strategy profile $\vec{a}$, if at each step some participant with a suboptimal strategy switches its strategy, the system converges to a pure Nash equilibrium in no more than $2n$ steps.*

Proof: Let $t = Cn/L$. For any strategy profile $\vec{a}$, consider the set $S_{\text{big}}(\vec{a})$ of "big" components of $G_{\vec{a}}$ of size greater than t and the set $S_{\text{small}}(\vec{a})$ of "small" components of $G_{\vec{a}}$ of size less than or equal to t. Define a potential function Φ by

$$\Phi(\vec{a}) = \sum_{A \in S_{\text{big}}(\vec{a})} |A| - \sum_{A \in S_{\text{small}}(\vec{a})} |A|.$$

It is easy to see that $-n \leq \Phi(\vec{a}) \leq n$ for any $\vec{a}$. We will now show that each step of the process reduces Φ by at least one. There are two main cases:

1. Some node i switches from insecure to secure. In this case i was previously an element of a component in S_{big} of size $m > t$. This former component becomes one or more new components with total size $m - 1$; if all of the resulting components are big, Φ is reduced by exactly one; otherwise, Φ is reduced by more than one as some components move from the positive to the negative side of the ledger.

2. Some node i switches from secure to insecure. In this case the resulting component containing i has $m \leq t$ elements, and it replaces one or more old components with total size $m - 1$. As both the new component and the old components are small, the net effect on Φ is to decrease it by one.

If each step reduces Φ by one, the number of steps must be less than the difference between the initial and final value of Φ, which is at most $n - (-n) = 2n$. ■

As a special case, we can start with $\vec{a} = 1^n$ and converge to an equilibrium from above by checking each node once. Each such test requires computing the size of the component in the attack graph, which takes time $O(|V| + |E|) = O(n^2)$ using depth-first search; this gives:

Corollary 31 *A Nash equilibrium can be computed in time $O(n^3)$.*

It is not hard to see that the $2n$ in Theorem 30 is close to the best possible bound, although a more careful analysis might reduce it slightly. A lower bound of n steps is trivial: in a system with $C < L/n$ and no players secure in the initial strategy profile, it takes n steps for all players to install the anti-virus software. To get closer to $2n$, consider a line with $t = \sqrt{n} - \frac{1}{2}$. Now consider an execution of the process where initially players 1 through $n - \sqrt{n}$, in increasing order, install to escape the single overlarge component; but then all players not at positions $k\sqrt{n}$ for some k uninstall; this takes $2n - 2\sqrt{n}$ steps.

We also have:

Corollary 32 *A pure Nash equilibrium always exists.*

6.3.3 Consequences of Changes in the Inoculation Cost

Though Theorem 29 suggests that we cannot hope to characterize the worst pure Nash equilibrium exactly, we can give a description of how it reacts to changes in the inoculation cost C.

Theorem 33 *The cost of the worst pure Nash equilibrium is a non-decreasing function of C when C ranges over $[2L/n, L)$.*

Proof: Fix some price of anti-virus software, $C \geq 2L/n$ so that $\lfloor Cn/L \rfloor \geq 2$. We shall use $\text{cost}(\vec{a}; C)$ to denote the cost of strategy profile $\vec{a}$ when the price is C.

Suppose we increase the price from C to $C' = C + \epsilon$ $(\epsilon > 0)$. We denote the worst-cost Nash equilibrium when the price is C by $\vec{a}$ and the worst-cost equilibrium when the price is C' by $\vec{b}$.

If the price increment is $\epsilon \leq L/n$, then the threshold (in Theorem 27) increases by at most one; that is, $\lfloor C'n/L \rfloor \leq \lfloor Cn/L \rfloor + 1$. We consider two cases:

Case 1: $\vec{a}$ is a Nash equilibrium for C'. This case is easy. Because $\vec{b}$ is a worst-cost Nash equilibrium for C', we have:

$$\text{cost}(\vec{a}; C) < \text{cost}(\vec{a}; C') \leq \text{cost}(\vec{b}; C').$$

Case 2: $\vec{a}$ is *not* a Nash equilibrium for C'. This can happen only if $\lfloor C'n/L \rfloor = \lfloor Cn/L \rfloor + 1$. Specifically, there must exist a node $w \in I_{\vec{a}}$ such that adding it into attack graph $G_{\vec{a}}$ yields a component of size $\lfloor Cn/L \rfloor$ but not $\lfloor C'n/L \rfloor$. Let us denote the sizes of components adjacent to w in G by $k_1, \ldots, k_s$.[2] We then have: $\sum_{i=1}^{s} k_i = \lfloor Cn/L \rfloor - 1$.

We define a new strategy $\vec{a}' = \vec{a}[w/0]$, which is the same as $\vec{a}$ except we no longer install anti-virus software on node w. Moreover,

[2]We say that a component $K \subseteq V$ is adjacent to node w if $\exists v \in K$ s.t. $(v, w) \in E$.

$$
\begin{aligned}
\operatorname{cost}(\vec{a'};C') - \operatorname{cost}(\vec{a};C) &= \frac{L}{n}\lfloor Cn/L\rfloor^2 - \left(C + \frac{L}{n}\sum_{i=1}^{s} k_i^2\right) \\
&\geq \frac{L}{n}\left(\lfloor Cn/L\rfloor^2 - \left(\sum_{i=1}^{s} k_i\right)^2\right) - C \\
&= \frac{L}{n}\left(\lfloor Cn/L\rfloor^2 - (\lfloor Cn/L\rfloor - 1)^2\right) - C. \quad (6.1)
\end{aligned}
$$

Equation (6.1) is non-negative whenever

$$2\lfloor Cn/L\rfloor - 1 \geq Cn/L,$$

which always holds by assumption for all $C \geq 2L/n$.

We repeat this process until there do not exist any nodes violating Nash equilibrium condition. At each step, the cost of our new strategy does not decrease. Therefore, if at the end we get a Nash equilibrium $\vec{d}$, then

$$\operatorname{cost}(\vec{a};C) \leq cost(\vec{a};C') \leq \operatorname{cost}(\vec{d};C') \leq cost(\vec{b};C').$$

Because we chose C arbitrarily, our argument holds for all values of ϵ.

■

6.3.4 Price of Anarchy

The notion of the **price of anarchy** was introduced by Koutsoupias and Papadimitriou in [KP99]. It is defined as the worst-case ratio between the cost of a Nash equilibrium and the cost of the optimal solution, and is thus a measure of how far

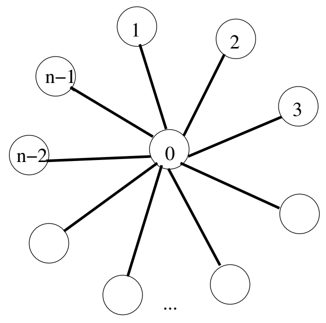

Figure 6.2: Star graph $G = K_{1,n}$ used in the proof of the lower bound.

away a Nash equilibrium can be from the social optimum.[3] When the network graph is G and the costs are C, L, we use $\rho(G, C, L)$ to denote the price of anarchy.

We show that, in our game, the price of anarchy is quite high, $\Theta(n)$. This is a consequence of two simple lemmas:

Lemma 34 *(Lower bound). Let G be the star graph $K_{1,n}$ (see Figure 6.2). Let the price of the anti-virus software be $C = L(n-1)/n$. Then*

$$\rho(G, C, L) = n/2.$$

Proof: The given C and L satisfy $t = Cn/L = n - 1$. From Corollary 28, it follows that installing anti-virus software on exactly one node is a Nash equilibrium. If pure Nash strategy $\vec{a}$ installs anti-virus software on some node that is not the center node, the cost will be $C + L(n-1)^2/n = L(n-1)$.

An optimal strategy for the star with the given C and L is $\vec{a}^* = (1, 0, \ldots, 0)$ (*i.e.*, only the center node installs anti-virus software.) Its cost is $C + L(n-1)/n = 2L(n-1)/n$.

[3]Because our game has a random component, the cost is an expected cost.

The price of anarchy is therefore

$$\frac{L(n-1)}{2L(n-1)/n} = \frac{n}{2}.$$

■

Lemma 35 *(Upper bound). Fix any graph G and costs C, L. Then*

$$\rho(G, C, L) \leq n.$$

Proof: Let $\vec{a}^*$ denote the optimum solution.

If $C > L$, no node in a Nash equilibrium will install anti-virus software. Hence, there is only one Nash equilibrium $\vec{a} = 0^n$, whose cost is Ln. If the optimum solution contains at least one secure node, then $\text{cost}(\vec{a}^*) \geq C > L$. (Otherwise, $\vec{a}^* = 0^n$ and $\rho(G, C, L) = 1$.) We thus have:

$$\rho(G, C, L) \leq \frac{Ln}{L} = n.$$

If $C \leq L$, then the expected cost of the worst Nash equilibrium $\vec{a}$ is at most Cn, because the expected cost to each node is at most C (if the expected cost to a node is greater than C, then it will want to switch to installing the software with probability 1.) If the optimum solution contains at least one secure node, then $\text{cost}(\vec{a}^*) \geq C$. Otherwise, the optimum solution contains no secure nodes and hence $\text{cost}(\vec{a}^*) \geq L \geq C$.

$$\rho(G, C, L) \leq \frac{Cn}{C} = n.$$

■

6.4 Optimization

Allowing users to selfishly choose whether or not to install anti-virus software may be grossly inefficient, relative to the social optimum. An alternative approach to this problem is for a benevolent dictator to attempt to maximize social welfare by centrally computing a solution and imposing it on all nodes.

Difficulties with this approach arise from the hardness of computing the optimum solution to the inoculation problem. In the first two sections, we give a characterization of the optimum solution and use it to show that the inoculation problem is **NP**-hard.

This suggests computing an approximate solution. We can find in polynomial time a solution with approximation ratio at most $O(\log^2 n)$; such a solution is substantially better than the $\Theta(n)$ ratio derived from the worst Nash equilibrium.

6.4.1 Characterization

We have a graph-theoretic characterization of optimum strategies, similar to our characterization of Nash equilibria in Theorem 27:

Theorem 36 *Fix C, L and let $t = Cn/L$. If $\vec{a}$ is an optimum strategy, then every component in attack graph $G_{\vec{a}}$ has size at most* $\max(1, (t+1)/2)$.

Proof: Strategy $\vec{a}$ partitions G into disjoint components. Pick some component $K \subseteq V$ from the attack graph, where $k = |K|$ is at least two. (If we can't find a component with at least two nodes, then all components in the attack graph have size one, and the theorem follows.)

If we install the anti-virus software on some node of K, we may get m new components in $G_{\vec{a}}$, where $0 \leq m \leq k - 1$. Let us denote the sizes of these new components by $k_1, \ldots, k_m$, where $\sum_{i=1}^{m} k_i = k - 1$. Because $\vec{a}$ is already an optimal strategy, installing the anti-virus software on an extra node cannot improve the total cost. Therefore, we have:

$$C + \frac{L}{n}\left(\sum_{i=1}^{m} k_i^2\right) \geq \frac{Lk^2}{n}$$

$$\Leftrightarrow$$

$$k^2 - \left(\sum_{i=1}^{m} k_i^2\right) \leq t. \tag{6.2}$$

If $m = 0$, then Equation (6.2) becomes:

$$k \leq \sqrt{t} \leq (t+1)/2.$$

Meanwhile, for $m > 0$,

$$\begin{aligned} k^2 - \left(\sum_{i=1}^{m} k_i^2\right) &\geq k^2 - \left(\sum_{i=1}^{m} k_i\right)^2 \\ &= k^2 - (k-1)^2 \\ &= 2k - 1. \end{aligned} \tag{6.3}$$

Combining Equations (6.2) and (6.3), we get:

$$k \leq (t+1)/2.$$

■

Unfortunately, the optimal solution is hard to compute.

Theorem 37 *It is **NP**-hard to compute an optimal strategy.*

Proof: The proof is by reduction from VERTEX COVER and is similar to the proof of Theorem 29. ■

6.4.2 Reduction to Sum-of-Squares Partition

Because it is unlikely that we can find an optimal solution, we naturally consider approximation algorithms.

The optimization problem that defines the inoculation problem can be posed as follows: choose the set of secure nodes $I_{\vec{a}}$ that minimizes the following objective function:

$$C|I_{\vec{a}}| + \frac{L}{n} \sum_{V \in \phi(I_{\vec{a}})} |V|^2,$$

where $\phi(I_{\vec{a}})$ is the set of connected components created by the removal of nodes in $I_{\vec{a}}$.

For the purposes of our approximation algorithm for the inoculation problem, we assume that we can guess $m = |I_{\vec{a}}|$, the number of secure nodes in an optimum configuration. This assumption is without loss of generality, because we can run our algorithm on all possible choices of $m = 1, \ldots n$ and take the best solution.

Thus, a solution to the inoculation problem is reduced to finding a solution to the problem of removing m nodes from a given graph to minimize the sum of the squares of the sizes of the surviving components. We discuss this problem in Section 6.5.

6.5 Sum-of-Squares Partitions

In Section 6.4.2, we came across the following problem, which we now analyze in more detail.

SUM-OF-SQUARES PARTITION PROBLEM: *By removing a set F of at most m nodes, partition the graph into disconnected components $H_1, \ldots, H_k$, such that $\sum_i |H_i|^2$ is minimum.*

Though we have arrived at this combinatorial optimization problem via our study of containing computer viruses, it may be of independent interest. Note that it inherits **NP**-hardness from the inoculation problem. The *edge version* of the sum-of-squares-partition problem is similar, but asks for the removal of m edges, rather than nodes, to disconnect the graph.

We prove the following theorem:

Theorem 38 *Let* OPT *be the optimum objective function value for the sum-of-squares partition problem on G with the removal of at most m nodes. We can find a set F of $O(\log^2 n)m$ nodes, such that their removal creates disconnected components $H_1, \ldots, H_l$ such that $\sum_i |H_i|^2 \leq O(1) \cdot \mathrm{OPT}$.*

Proof: Details can be found in the Yale CS technical report [ACY04]. ▮

An immediate consequence of this theorem is the existence of an approximation algorithm for the inoculation problem:

Corollary 39 *If* OPT *is the cost of the optimum solution for the inoculation problem, there exists a polynomial-time approximation algorithm that finds a solution with cost at most $O(\log^2 n) \cdot \mathrm{OPT}$.*

Proof: Suppose an optimum solution contains m secure nodes, and the sizes of the insecure node components are $k_1, \ldots, k_p$, so that $\mathrm{OPT} = Cm + L/n \sum_i k_i^2$. Using our approximation algorithm for the sum-of-squares partition problem, we can find a set of $O(\log^2 n)m$ secure nodes such that the sum of the squares of the corresponding insecure components is at most $O(1)\sum_i k_i^2$. Thus, the cost of the approximate solution is:

$$\begin{aligned}
O(\log^2 n)\cdot Cm + O(1)\cdot \frac{L}{n}\sum_i k_i^2 &\leq O(\log^2 n)\cdot Cm + O(\log^2 n)\cdot \frac{L}{n}\sum_i k_i^2 \\
&= O(\log^2 n)\cdot \mathrm{OPT}.
\end{aligned}$$

■

6.6 Related Work

In this section, we describe three classes of work related to this chapter: virus propagation models, economic models of investment in security, and game-theoretic models of security. We then discuss some work on the graph partition problem that is related to the sum-of-squares partition problem we consider in Section 6.5.

6.6.1 Virus Propagation Models

Traditional epidemiological models characterize the viral infection in terms of birth rate and death rate of the virus [Bai75, Fra80]. Usually, these models assume that an infected individual is equally likely to infect any other individual in the population; in contrast, computer viruses usually spread via localized interactions. Kephart and White extended the traditional model by transferring it onto a directed random graph [KW91]. Later work (*e.g.*, [KCW93, KW93, WKE00]) studied virus propagation over other kinds of graphs, including Internet-like power-law graphs [PSV02a, PSV02b, ZTG02]. We do not restrict the network topology in any way and consider a general undirected graph. Our model is in some ways closer to models in percolation theory (see [Kes82]): an infected node infects all of its unprotected neighbors, spreading infection throughout the graph until it is blocked by an anti-virus software.

6.6.2 Economic Models of Security

Our work is motivated in part by an observation that security technologies exhibit network externalities [And01]. Specifically, the benefit obtained by using security technology (anti-virus software in our case) does not accrue only to the user of the security technology but rather to all users of the network. Aspnes *et al.* [AFMP03] also consider anti-virus immunization, and proposed studying how to encourage highly connected nodes to use anti-viral techniques.

We assume that costs of installation and infection are known. Alternatively, one could use risk analysis to estimate the costs and benefits from installing a security technology (see, for example, [Hoo00]), or estimate values based on empirical studies of the costs of security breaches [CGLZ03, GL02].

6.6.3 Game-Theoretic Models of Security

Application of game theory to network security has yielded interesting results [HMOS02b, HMOS02a,Syv97]. For example, Bell uses a simple game to study network reliability. In the game, the router tries to find a least cost path and a network tester tries to maximize this cost by failing links [Bel01]. Kunreuther and Heal recently introduced the notion of **interdependent security** (IDS) games, in which decisions to adopt security technology by one agent affect other agents [KH03]. Kearns and Ortiz subsequently extended their paper and gave an algorithm for finding approximate Nash equilibria in this model [KO04].

Our work is similar to work on IDS games in certain respects: each agent in both our game and an IDS game makes a decision whether or not to invest money in a security technology, and this decision affects other agents. The main differences are that we assume that installing anti-virus software protects against all bad effects of viruses, while the IDS work concentrates on negative side-effects of security breaches even on protected parties; and we assume a restricted network topology that contains the spread of viruses, while the IDS work assumes a complete topology.

6.6.4 Graph Partition Problems

In Section 6.5, we describe and provide an approximate solution for a new graph partitioning problem. Previous work on other forms of graph partitioning includes the approximation algorithm of Leighton and Rao [LR99] for **sparsest cut**, from which the same authors derive a pseudo-approximation algorithm for **b-balanced cuts**, where each side of the cut must have size $b|V|$ or greater. Arora *et al.* [ARV04] recently improved the approximation ratios of these results. The case of $b = 1/2$ is **graph bisection**, for which Feige and Krauthgamer [FK02] give a good approximation algorithm. Even *et al.* [ENRS99] give $O(\log n)$-ratio pseudo-approximation algorithms for several balanced partitioning problems, including the **ρ-separator problem** and the **k-balanced partitioning problem**.

Chapter 7

Conclusions and Open Questions

We conclude by summarizing our contributions and discussing some problems left open in our research.

7.1 Conclusions

The thesis of this dissertation is that **secure distributed systems can be built given the right set of tools**. Our results support this thesis. We have provided a set of practical tools, which solve sundry security problems in a distributed setting. We have also examined how the overall security of distributed systems is affected by security choices of their participants.

In Chapter 3, we defined and constructed a blind coupon mechanism (BCM), implementing a specialized form of an AND-homomorphic bit commitment. The BCM has many natural applications such as tallying votes over the Internet and short discrete proofs of circuit satisfiability. In particular, it can be used to spread an alert undetectably in a variety of epidemic-like settings despite the existence of Byzantine nodes and a powerful, active adversary.

Then in Chapter 4, we presented a simple and efficient construction of a verifiable random function (VRF). Our VRF's proofs and keys consist of only one group element regardless of the input size. To this date, this is the only VRF (and PRF) in the standard model, which does not process its inputs bit-by-bit. We demonstrated that our scheme can be instantiated with elliptic groups of very reasonable size which makes our constructions quite practical. Our VRF also possesses many useful algebraic properties: It admits zero-knowledge proofs of knowledge of the VRF value. It is also possible to obtain the VRF value on committed inputs. Our VRF has already been used by other researchers in constructing a compact e-cash system [CHL05] and a distributed electronic lottery [CHYC05].

In Chapter 5, we designed the first reasonably efficient threshold and proactive pseudo-random permutation (PRP). Our scheme requires $O(1)$ communication rounds and low computational overhead, which is a substantial advantage over the generic multi-party solutions. PRPs are commonly used tools in protocol design. Our techniques enable distributing many protocols (using PRPs), which until now only existed in the centralized setting. In particular, we showed how to distribute a remotely keyed authenticated encryption scheme, a CCA-secure cryptosystem, and a CBC encryption mode.

Finally, in Chapter 6, we described a simple economic game that represents the difficult problem of choosing on which nodes to install anti-virus software to contain the spread of computer viruses in a network. The Nash equilibria of this game have a simple characterization, and we proved that in the worst case, the ratio between the social cost of a Nash equilibrium and a social optimum can be linear in the number of nodes. We have also shown how a near-optimal deployment of anti-virus software can be computed by reduction to the **sum-of-squares partition problem**, a new variant of classical graph partitioning problems. We gave a polynomial-time $O(\log^2 n)$-approximation algorithm for sum-of-squares partition, which yields a corresponding approximation algorithm for anti-virus software deployment. This algorithm may be of use as a network administration tool for choosing how to deploy anti-virus software to minimize the combined costs of deployment and infection.

7.2 Open Problems

We now list some open problems related to the results in this dissertation.

Security of our cryptographic tools rests on the difficulty of several computational problems. In particular, we showed that our VRF and threshold PRP constructions are secure as long as the y-DDHI problem (given $(g, g^x, \ldots, g^{(x^y)}), R)$, decide if $R = g^{1/x}$) is hard on elliptic curve groups. We also proved that our BCM scheme is secure if it is hard to find points on elliptic curves over $\mathbb{Z}_n$ (for $n = pq$ of unknown factorization). While our new assumptions certainly appear reasonable, reducing them to widely-believed simpler assumptions is of great value. A natural question is whether we can obtain our tools (BCM, VRF, and threshold PRP) under established intractability assumptions (such as factoring or computing discrete logs) without which most of cryptography collapses anyway.

Our BCM scheme (Chapter 3) allows undetectably transmitting only one bit across the network. We described several ways to expand the bandwidth of the system by using multiple blind coupon schemes in parallel: We can encode a message by a vector of coupons over different moduli, where several coupons represent each message bit. However, even with an efficient encoding, we have to construct and run $\Omega(n)$ blind coupon mechanisms in parallel to transmit n bits. What's worse, once we have transmitted n bits, there is no mechanism to reset the communication system to allow new messages to be transmitted, and any communication system that could be reset is vulnerable to replay attacks by the faulty nodes. It remains an open problem whether we can use a BCM to transmit multiple bits without a linear blow-up in the message size. We could solve this problem with a signed encryption scheme $\mathcal{E}(\cdot)$ that is homomorphic with respect to the max operation. In this case, message could be transmitted one bit at a time with a simple acknowledgment protocol by having the sender signal that $b_i = b$ by transmitting $\mathcal{E}(3i+b)$ and having the receiver respond as each bit is received by transmitting $\mathcal{E}(3i+2)$. Each node would propagate the max of all messages it had previously received, which it would compute blindly using the combining function. The rather large difficulty is in finding suitable max-homomorphic encryption and signing functions.

Our VRF construction (Chapter 4) is based on the y-DDHI assumption of high polynomial order $y = 2^{a(k)}$, where $a(k)$ is the input size. This allows us to support small inputs of slightly superlogarithmic size $a(k) = \omega(\log k)$, while longer inputs have to be processed by a collision-resistant hash function. It is an open question if we can further reduce the order by mapping inputs to codewords of an error-correcting code as in [Dod03, Lys02].

Our protocol for threshold and proactive PRP (Chapter 5) is secure only against semi-honest attackers, who do not try to disrupt the computation. It makes sense to secure it against malicious attackers by using standard techniques such as verifiable secret sharing and zero-knowledge proofs [RBO89]. However, this extension seems to be very messy. It is also open whether we can use group multiplication to implement the distributed Feistel transformation rather than having to convert group elements into bit strings. This would allow us to avoid altogether the use of an expensive protocol by Damgård *et al.* [DI05], greatly improving our construction's efficiency.

Finally, our economic model of network security (Chapter 6) makes some very strong simplifying assumptions: every infected node eventually infects all unprotected neighbors; the costs of installing the anti-virus software and becoming infected are known and equal for all nodes; the virus imposes no costs on protected nodes; and nodes can observe which of the other nodes intend to install the anti-virus software and adjust their own strategies in response. None of these assumptions correspond completely to reality, but we believe that as a first step the resulting model is a reasonable compromise between accuracy and analyzability, and that the results obtained with the model (especially the characterization of Nash equilibria) are similar to what one might expect with a more complex model that took into account limited information and learning by individual nodes. The natural next step is to incorporate more details in the model and see if such changes affect the results; this might involve both theoretical work to predict the effect of changes and experimental or observational work to study how real-world decision-makers choose whether or not to deploy specific security mechanisms.

Bibliography

[Abe99] Masayuki Abe. Mix-networks on permutation networks. In *Advances in Cryptology - Proceedings of ASIACRYPT 99*, volume 1706 of *Lecture Notes in Computer Science*, pages 258–273. Springer-Verlag, 1999.

[ACS02] Joy Algesheimer, Jan Camenisch, and Victor Shoup. Efficient computation modulo a shared secret with applications to the generation of shared safe prime products. In *Advances in Cryptology - Proceedings of CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 417–432. Springer-Verlag, 2002.

[ACY04] James Aspnes, Kevin Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. Technical Report YALEU/DCS/TR-1295, Yale University, July 2004. Available at `ftp://ftp.cs.yale.edu/pub/TR/tr1295.pdf`.

[ACY05] James Aspnes, Kevin Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. In *Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 43–52, 2005.

[ADG$^+$05] James Aspnes, Zoë Diamadi, Kristian Gjøsteen, René Peralta, and Aleksandr Yampolskiy. Spreading alerts quietly and the subgroup escape problem. In *Advances in Cryptology - Proceedings of ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 253–272, 2005.

[AFMP03] James Aspnes, Joan Feigenbaum, Michael Mitzenmacher, and David Parkes. Towards better definitions and measures of Internet security. In *Workshop on Large-Scale Network Security and Deployment Obstacles*, 2003.

[And01] Ross Anderson. Why information security is hard - an economic perspective, 2001. Available at `http://www.cl.cam.ac.uk/~rja14/econsec.html`.

[ARV04] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, pages 222–231, 2004.

[Bac85] Eric Bach. *Analytic Methods in the Analysis and Design of Number-Theoretic Algorithms.* A.C.M. Distinguished Dissertations. MIT press, Cambridge, MA, 1985.

[Bai75] Norman T. Bailey. *The Mathematical theory of infectious diseases and its applications.* Hafner Press, 1975.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology - Proceedings of EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology - Proceedings of EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCF00] Ernest F. Brickell, Giovanni Di Crescenzo, and Yair Frankel. Sharing block ciphers. In Ed Dawson, Andrew Clark, and Colin Boyd, editors, *ACISP*, volume 1841 of *Lecture Notes in Computer Science*, pages 457–470. Springer, 2000.

[BD01] Amos Beimel and Shlomi Dolev. Buses for anonymous message delivery. In *Second International Conference on FUN with Algorithms*, pages 1–13. Carleton Scientific, 2001.

[Bel01] Michael G. H. Bell. The measurement of reliability in stochastic transport networks. In *Proceedings of 2001 Intelligent Transportation Systems*, pages 1183–1188, 2001.

[BF01a] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *Lecture Notes in Computer Science*, 2139:213–229, 2001.

[BF01b] Dan Boneh and Matthew K. Franklin. Efficient generation of shared RSA keys. *Journal of the Association for Computing Machinery*, 48(4):702–722, 2001.

[BFN98] Matt Blaze, Joan Feigenbaum, and Moni Naor. A formal treatment of remotely keyed encryption. In *Advances in Cryptology - Proceedings of EUROCRYPT 98*, Lecture Notes in Computer Science, pages 251–265. Springer-Verlag, 1998.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of Second Theory of Cryptography Conference (TCC 2005)*, pages 325–341, 2005.

[BIB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds. In *Proceedings of the ACM Symposium on Principles of Distributed Computation*, pages 201–209, 1989.

[BLZ94] Johannes A. Buchmann, J. Loho, and J. Zayer. An implementation of the general number field sieve. *Lecture Notes in Computer Science*, 773:159–166, 1994.

[BoGW88] Michael Ben-or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computing. In *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

[BS02] Dan Boneh and Alice Silverberg. Application of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080, 2002. `http://eprint.iacr.org/2002/080/`.

[BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BSS99] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1999.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42th IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[CGH00] Dario Catalano, Rosario Gennaro, and Shai Halevi. Computing inverses over a shared secret modulus. In *Advances in Cryptology - Proceedings of EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 190–206. Springer-Verlag, 2000.

[CGLZ03] Katherine Campbell, Lawrence A. Gordon, Martin P. Loeb, and Lei Zhou. The economic cost of publicly announced information security breaches: Empirical evidence from the stock market. *Journal of Computer Security*, 2003.

[Cha81] David Chaum. Untraceable electronic mail, return address and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - Proceedings of CRYPTO 82*, pages 199–203, 1982.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *Advances in Cryptology - Proceedings of EUROCRYPT 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005. to appear.

[CHYC05] Sherman S. M. Chow, Lucas Chi Kwong Hui, Siu-Ming Yiu, and K. P. Chow. An e-lottery scheme using verifiable random function. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *ICCSA (3)*, volume 3482 of *Lecture Notes in Computer Science*, pages 651–660. Springer, 2005.

[CKPS01] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology - Proceedings of CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 524–541. Springer-Verlag, 2001.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology - Proceedings of CRYPTO 2004*, pages 56–72, 2004.

[COS86] Don Coppersmith, Andrew M. Odlyzko, and Richard Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1(1):1–15, 1986.

[CRS04] David Chaum, Peter Y.A. Ryan, and Steve A. Schneider. A practical, voter-verifiable election scheme. Technical Report CS-TR-880, School of Computing Science, University of Newcastle, December 2004.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Proceedings of EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer-Verlag, 2002.

[DA03] Yevgeniy Dodis and Jee Hea An. Concealment and its applications to authenticated encryption. In *Advances in Cryptology - Proceedings of EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 312–329. Springer-Verlag, 2003.

[Dam87] Ivan Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology - Proceedings of EUROCRYPT 87*, Lecture Notes in Computer Science, pages 203–216. Springer-Verlag, 1987.

[Dem93] N. Demytko. A new elliptic curve based analogue of RSA. In *Advances in Cryptology - Proceedings of EUROCRYPT 93*, volume 765 of *Lecture Notes in Computer Science*, pages 40–49. Springer-Verlag, 1993.

[Des87] Yvo Desmedt. Society and group-oriented cryptography: a new concept. In *Advances in Cryptology - Proceedings of CRYPTO 87*, pages 120–127, 1987.

[Des97] Yvo Desmedt. Some recent research aspects of threshold cryptography. In *First International Workshop On Information Security*, pages 158–173, 1997.

[Des00] Anand Desai. New paradigms for constructing symmetric encryption schemes secure against chosen-ciphertext attack. In *Advances in Cryptology - Proceedings of CRYPTO 2000*, pages 394–412, 2000.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology - Proceedings of CRYPTO 89*, pages 307–315, 1989.

[DF91] Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures. In *Advances in Cryptology - Proceedings of CRYPTO 91*, pages 457–469, 1991.

[DFK$^+$06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Third Theory of Cryptography Conference*, 2006. To appear.

[DGH$^+$87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In Fred B. Schneider, editor, *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 1–12, Vancouver, BC, Canada, August 1987. ACM Press.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in*

Cryptology - Proceedings of CRYPTO 2005, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394. Springer-Verlag, 2005.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Fourth International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136, 2001.

[Dod03] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In *Proceedings of 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 1–17, 2003.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Eighth International Workshop on Theory and Practice in Public Key Cryptography*, pages 416–431, 2005.

[DYY06] Yevgeniy Dodis, Aleksandr Yampolskiy, and Moti Yung. Threshold and proactive pseudo-random permutations. In *Third Theory of Cryptography Conference (TCC 2006)*, volume 3876 of *Lecture Notes in Computer Science*, pages 542–560, 2006.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[ENRS99] Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Fast approximate graph partitioning algorithms. *SIAM Journal on Computing*, 28:2187–2214, 1999.

[FK02] Uriel Feige and Robert Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31:1090–1118, 2002.

[Fra80] James C. Frauenthal. *Mathematical modeling in epidemiology.* Springer-Verlag, New York, 1980.

[Gal01] Steven D. Galbraith. Supersingular curves in cryptography. *Lecture Notes in Computer Science*, 2248:495–513, 2001.

[Gal02] Steven D. Galbraith. Elliptic curve Paillier schemes. *Journal of Cryptology*, 15(2):129–138, 2002.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33:792–807, 1986.

[GJ04] Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Advances in Cryptology - Proceedings of EUROCRYPT 2004*, pages 456–473, 2004.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology - Proceedings of EUROCRYPT 99*, pages 295–310, 1999.

[GJKR01] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. *Inf. Comput.*, 164(1):54–84, 2001.

[Gjø04] Kristian Gjøsteen. *Subgroup membership problems and public key cryptosystems*. PhD thesis, NTNU, May 2004.

[Gjø05] Kristian Gjøsteen. Symmetric subgroup membership problems. In Serge Vaudenay, editor, *Proceedings of Public Key Cryptography 2005*, volume 3386 of *LNCS*, pages 104–119. Springer-Verlag, 2005.

[GK99] Shafi Goldwasser and Joe Kilian. Primality testing using elliptic curves. *Journal of the Association for Computing Machinery*, 46:450–472, 1999.

[GL89] Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21th Annual ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[GL02] Lawrence A. Gordon and Martin P. Loeb. The economics of information security investment. *ACM transactions on information and system security*, pages 438–457, 2002.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing*, pages 270–299, 1982.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[Gol98] Oded Goldreich. Secure multi-party computation, 1998. Available from `http://www.wisdom.weizmann.ac.il/~oded/foc.html`.

[Hal04] Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In *Advances in Cryptology - Proceedings of INDOCRYPT 2004*, pages 315–327, 2004.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology - Proceedings of CRYPTO 95*, pages 339–352, 1995.

[HMOS02a] Samuel N. Hamilton, Wendy L. Miller, Allen Ott, and O. Sami Saydjari. Challenges in applying game theory to the domain of information warfare. In *4th Information survivability workshop (ISW-2001/2002)*, Vancouver, Canada, 2002.

[HMOS02b] Samuel N. Hamilton, Wendy L. Miller, Allen Ott, and O. Sami Saydjari. The role of game theory in information warfare. In *4th Information survivability workshop (ISW-2001/2002)*, Vancouver, Canada, 2002.

[Hoo00] Kevin Hoo. How much is enough? A risk-management approach to computer security. Consortium for Research on Information Security Policy (CRISP) Working Paper., 2000.

[HR03] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In *Advances in Cryptology - Proceedings of CRYPTO 2003*, pages 482–499, 2003.

[HWL87] Jr. Hendrik W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[Jak98] Markus Jakobsson. A practical Mix. In *Advances in Cryptology - Proceedings of EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*, pages 448–461. Springer-Verlag, 1998.

[Jak99] Markus Jakobsson. Flash mixing. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 83–89. ACM, 1999.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In *CT-RSA*, pages 244–262, 2002.

[JMV02] Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Blockwise-adaptive attackers. revisiting the (in)security of some provably secure encryption modes: CBC, GEM, IACBC. In *Advances in Cryptology - Proceedings of CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 17–30. Springer-Verlag, 2002.

[JN01] Antoine Joux and Kim Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/2001/003/`.

[JS04] Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing big brother : an abuse-resilient transaction escrow scheme. In *Advances in Cryptology - Proceedings of EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 590–608. Springer-Verlag, 2004.

[KCW93] Jeffrey O. Kephart, David M. Chess, and Steve R. White. Computers and epidemiology. In *IEEE Spectrum*, pages 20–26, 1993.

[Kes82] Harry Kesten. *Percolation Theory for Mathematicians*, volume 2. Birkhäuser, Boston, 1982.

[KH03] Howard Kunreuther and Geoffrey Heal. Interdependent security. *Journal of Risk and Uncertainty (Special Issue on Terrorist Risks)*, 2003.

[Kil05] Eike Kiltz. Uncoditionally secure constant round multi-party computation for equality, comparison, bits and exponentiation. Cryptology ePrint Archive, Report 2005/066, 2005.

[KK98] Noboru Kunihiro and Kenji Koyama. Equivalence of counting the number of points on elliptic curve over the ring Z_n and factoring n. In Nyberg [Nyb98].

[KMOV92] Kenji Koyama, Ueli M. Maurer, Tatsuaki Okamoto, and Scott A. Vanstone. New public-key schemes based on elliptic curves over the ring z_n. In *Advances in Cryptology - Proceedings of CRYPTO 91*, volume 576 of *Lecture Notes in Computer Science*, pages 252–266, 1992.

[KO04] Michael Kearns and Luis Ortiz. Algorithms for interdependent security games. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[KP99] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th annual symposium on theoretical aspects of computer science*, pages 403–413, 1999.

[KW91] Jeffrey O. Kephart and Steve R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Symposium on Security and Privacy*, pages 343–361, 1991.

[KW93] Jeffrey O. Kephart and Steve R. White. Measuring and modeling computer virus prevalence. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1993.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal of Computing*, 17:373–386, 1988.

[LR99] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the Association for Computing Machinery*, 46(6):787–832, 1999.

[LSP82] Leslie Lamport, Robert Shostack, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Proggramming Languages and Systems*, 4(3):382–401, 1982.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from DH-DDH separation. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 597–612, 2002.

[LZ94] Georg-Johann Lay and Horst G. Zimmer. Constructing elliptic curves with given group order over large finite fields. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, pages 250–263. Springer-Verlag, 1994.

[Mau94] Ueli M. Maurer. Towards proving the equivalence of breaking the diffie-hellman protocol and computing discrete logarithms. In *Advances in Cryptology - Proceedings of CRYPTO 94*, Lecture Notes in Computer Science, pages 271–281, 1994.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. *Lecture Notes in Computer Science*, 2139:542–565, 2001.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In *CT-RSA*, pages 149–163, 2002.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 120–130, 1999.

[MSK02] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, pages 481–484, 2002.

[MSNWW05] Keith M. Martin, Rei Safavi-Naini, Huaxiong Wang, and Peter R. Wild. Distributing the encryption and decryption of a block cipher. *Designs, Codes, and Cryptography*, 2005. to appear.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography.* CRC press LLC, Boca Raton, FL, 1997.

[NBD01] Juan Manuel González Nieto, Colin Boyd, and Ed Dawson. A public key cryptosystem based on the subgroup membership problem. In S. Quing, T. Okamoto, and J. Zhou, editors, *Proceedings of ICICS*

2001, volume 2229 of *Lecture Notes in Computer Science*, pages 352–363. Springer-Verlag, 2001.

[Nie03] Jesper Buus Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology - Proceedings of CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 401–416. Springer-Verlag, 2003.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In *Advances in Cryptology - Proceedings of EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer-Verlag, 1999.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 458–467, 1997.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In Nyberg [Nyb98], pages 308–318.

[Nyb98] Kaisa Nyberg, editor. *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[OSS84] H. Ong, Claus-Peter Schnorr, and Adi Shamir. An efficient signature scheme based on quadratic equations. In *proceedings of ACM Symposium on Theory of Computing*, ACM, pages 208–216, 1984.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology - Proceedings of EUROCRYPT 98*, volume 1403, pages 308–318. Springer-Verlag, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residue classes. In Jacques Stern, editor, *Proceedings of EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology - Proceedings of EUROCRYPT 91*, pages 522–526, 1991.

[Pol75] John M. Pollard. A Monte Carlo method for factorization. *BIT*, 15:331–334, 1975.

[PRS04] Sarvar Patel, Zulfikar Ramzan, and Ganapathy S. Sundaram. Efficient constructions of variable-input-length block ciphers. In *Selected Areas in Cryptography 2004*, pages 326–340, 2004.

[PSV02a] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemics and immunization scale-free networks. In S. Bornholdt and H.G. Schuster, editors, *Handbook of graphs and networks: from the genome to the Internet*, pages 113–132, Berlin, 2002. Wiley-VCH.

[PSV02b] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. In *Physical Review Letters*, volume 65, 2002.

[Rab79] Michael Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology, January 1979.

[Rab98] Tal Rabin. A simplified approach to threshold and proactive RSA. In *Advances in Cryptology - Proceedings of CRYPTO 98*, pages 89–104, 1998.

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21th Annual ACM Symposium on the Theory of Computing*, pages 73–85, 1989.

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27:701–717, 1980.

[SGR98] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and Onion routing. *IEEE Journal on Selected Areas in Communications: Special Issue on Copyright and Privacy Protection*, 16(4):482–494, 1998.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Proceedings of EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1997.

[Sil99] Joseph H. Silverman. Computing rational points on rank 1 elliptic curves via L-series and canonical heights. *Mathematics of computation*, 68(226):835–858, April 1999.

[SP87] Claus P. Schnorr and John M. Pollard. An efficient solution of the congruence $x^2+ky^2 \equiv m \pmod{n}$. *IEEE Transactions on Information Theory*, 33(5):702–709, 1987.

[SRG00] Paul F. Syverson, Michael G. Reed, and David M. Goldschlag. Onion routing access configurations. In *DISCEX2000:Proceedings of the DARPA information survivability conference and exposition*, pages 34–40. IEEE CS Press, 2000.

[STW96] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 31–37, 1996.

[Syv97] Paul F. Syverson. A different look at secure distributed computation. In *IEEE Computer Security Foundations Workshop (CSFW 10)*, pages 109–115. IEEE Computer Society Press, June 1997.

[WKE00] Chenxi Wang, J. C. Knight, and M. C. Elder. On computer viral infection and the effect of immunization. In *ACSAC*, pages 246–256, 2000.

[Yao82] Andrew Yao. Protocols for secure computation (extended abstract). In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[ZTG02] Cliff Zou, Dan Towsley, and Weibo Gong. Email virus propagation modeling and analysis. Technical Report CSE-03-04, University of Massachusetts, Amherst, 2002.