

APPLICATIONS OF CAYLEY GRAPHS, BILINEARITY,
AND HIGHER-ORDER RESIDUES TO CRYPTOLOGY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jeremy Aaron Horwitz
September 2004

© Copyright by Jeremy Aaron Horwitz 2005
All Rights Reserved

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Boneh
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

John Mitchell

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ramarathnam Venkatesan

Approved for the University Committee on Graduate Studies.

Preface

We discuss three main topics: the use of Cayley graphs to present an essentially optimal algorithm for the discrete logarithm problem, the extension of identity-based encryption (IBE) to hierarchical IBE, and the application of higher-order residues to various cryptologic applications.

First, we focus on the discrete-logarithm problem, showing an algorithm that works in optimal time (up to logarithmic factors) and uses only a small amount of space. Our algorithm is a modification of the classic Pollard rho algorithm, introducing explicit randomization of the parameters for the updating steps of the algorithm. In proving that the algorithm works as claimed, we see several intermediate and related results of independent interest.

Next, we present the concept of hierarchical identity-based encryption (HIBE) schemes. An IBE scheme is one in which any string can be used as a public key (*e.g.*, the recipient's e-mail address); private keys are distributed by one central authority. An HIBE scheme is the hierarchical analogue of an IBE; among its many advantages those that come from splitting the key-distribution burden as well as those that come from escrow-related applications. We present an example of an HIBE scheme, give security definitions, and mention some of the applications of HIBE schemes.

Finally, we describe some cryptologic applications of the r th-power-residue symbol (a higher-order analogue of the Jacobi symbol). We present an encryption scheme in which recipients only need an r th root of unity to decrypt. By simply changing the value of r and distributing roots to a new group, controlling who receives such roots, the set of decrypters can be changed without generating a new modulus ($N = pq$). We present other applications of the system as well as another application of the

*r*th-power-residue symbol: how to speed up the elliptic curve method for factoring numbers of the form $N = pq^r$.

Acknowledgements

I have owe a great deal of thanks and have a great deal of respect for my coauthors Dan Boneh, Ben Lynn, and Ramarathnam “Venkie” Venkatesan. Without them and their invigorating discussions, graduate school would have much more difficult and much less interesting.

Further thanks go to Dan who has also been my advisor; thanks to him for his support, motivation, and guidance. Dan is both an oracle (especially of cryptographic knowledge) and an easy-to-work-with person — a rare combination. Whenever I would head in the wrong direction, whenever I had a question (about graduate school in general or some technical issue), Dan had the answer and pushed me in the right direction. Simply put, I could not have hoped for a better advisor.

So much of the research in here was helped by long conversations and lengthy e-mail correspondence with Venkie. It is an understatement to say that he saved the day time and again. Also, if pop-cultural knowledge and a sense of humor were more important, he would be the greatest mind in cryptography rather than just one of the best.

Further thanks go to all my roommates over the past few years who have had to endure me working on this. Special thanks to Jon and Glenn for answering so many technical questions, without whose answers I would probably still be writing this dissertation.

So many friends deserve thanks, especially those who helped me with anything even vaguely dissertation-related, whether it was checking the technical details of a proof, proofreading, or even bringing me food so I could continue working uninterrupted. Also, thanks for putting up with my disappearance from social activity as

the end of the work on this dissertation drew nearer.

Finally, I want to thank my family for their support and faith through the years: my parents, my sister, and my grandmother, whom I know is smiling down on me as I write this.

Contents

Preface	iv
Acknowledgements	vi
1 Introduction	1
1.1 Background	1
1.2 Hardness Assumptions	2
1.3 Overview of Results	3
2 Random Cayley Graphs and Discrete Logarithms	7
2.1 Introduction	7
2.2 Preliminaries and Statement of Results	10
2.2.1 Cayley Digraphs	10
2.2.2 Limited Independence	12
2.2.3 Finding Cycles in Succinct Graphs and DLOG	13
2.2.4 The Basic Walk and the Natural Walk	15
2.2.5 Notational Conventions	16
2.2.6 Our Results	17
2.2.7 The Pairwise Independence of ϕ_S	18
2.3 The Basic Walk	23
2.3.1 Previous Work	24
2.3.2 The Markov Chain Induced by G	25
2.3.3 Rapid Mixing (Basic Walk, Total Independence)	27
2.3.4 Pseudorandomness	35

2.4	Proof of Theorem 2.2.4 (Degeneracy)	35
2.5	Proof of Theorem 2.2.5 (Cycle-Finding Time)	38
2.6	Secure Hash Functions	42
2.7	Experimental Results	44
3	Identity-Based Encryption from Bilinear Maps	46
3.1	Introduction	46
3.2	Definitions	49
3.2.1	Security	51
3.3	An HIBE Scheme Resistant against Domain Collusion	53
3.3.1	The BDH Assumption	53
3.3.2	A Game Transformation	54
3.3.3	Linear e -One-Way Functions	55
3.3.4	Our Domain-Collusion Resistant Scheme	58
3.3.5	Proof of Security	59
3.4	The Gentry-Silverberg HIBE Scheme	66
4	Weak Trapdoors from rth-Power Residues	68
4.1	Introduction	68
4.2	The r th-Power-Residue Symbol	70
4.2.1	An Alternate Algorithm for Computing r th-Residue Symbols	72
4.3	The Hardness of Distinguishing r th Residues	73
4.4	A Cryptosystem with a Reusable Modulus	75
4.4.1	Partial Decryption	78
4.4.2	Encrypted and Signed Messages	80
4.5	Factoring Integers of the Form $N = pq^r$	81
4.6	Open Problems	83
4.7	Conclusions	84
	Bibliography	85

Chapter 1

Introduction

1.1 Background

Cryptology is the scientific study of writing secret information. Throughout history, people have been interested in keeping information secret from certain people — and people have been interested in reading secret information that is being kept from them. Cryptology has two main branches: cryptography, the study of designing methods for hiding information (generally, from all but a select group of people), and cryptanalysis, the study of methods for getting at hidden information.

The most fundamental cryptographic problem is designing a pair of algorithms (called an *encryption scheme*): one for turning regular text into secret text (called an *encryption algorithm*) and one for undoing the encryption, turning secret text back into its original text (this algorithm is called a *decryption algorithm*). These algorithms should have the property that the decryption can only be done by people whom the *sender* (the person doing the encrypting) intended to be able to read it (the *receivers*).

A classical example of such an algorithm is the shift cipher: before sending any messages, the sender somehow shares a number between 1 and 25 (represent this number by the variable k) with the receivers. Then, the encryption algorithm simply takes the message, letter by letter, and shifts it k letters forward in the alphabet (wrapping around from Z to A). The decryption algorithm simply shifts backwards

by k . For example, if Alice and her friend Bob both shared $k = 3$, Alice would encrypt the message ATTACKATDAWN as DWWDFNDWGDZQ. Bob, knowing $k = 3$, could decrypt the message and learn when Alice wishes to attack. Additionally, Eve, a third party who doesn't know the value of K , will simply see DWWDFNDWGDZQ.

A natural cryptanalytic problem goes along with the cryptographic problem of designing an encryption scheme: a cryptanalyst can try to devise a decryption algorithm that allows an unintended (by the encrypter) recipient to recover an encrypted message (this is referred to as *breaking* the encryption scheme). In our example, (assuming the initial message is in English), a cryptanalyst could simply try all 25 values for k and then see which message is English. As long as the message is long enough (generally just a few characters are all that is needed), only one k will work and, thus, the cryptanalyst can read the original message, even though this is exactly what the encrypter sought to avoid.

In theory, most encryption schemes can be broken; however, there is one major issue in converting this theory to practice: time. Breaking the shift cipher is clearly easy when English text (or text in any standard spoken language) is sent, which is why this cipher is not used in any real applications. However, a more advanced encryption system for which a cryptanalyst will likely require 10,000 years to break is not generally considered breakable in the real world (only in theory).

Over the centuries, as cryptanalysts found ways to break cryptographers schemes, cryptographers had to create newer, more robust schemes. As computers became prevalent, cryptographers were able to create more advanced encryption schemes; this boon was not lost on cryptanalysts, who also began to use computers to their advantage.

1.2 Hardness Assumptions

Rather than proving that an encryption scheme is hard to break, most security results are relative; that is, they show that you could only break the scheme being discussed if you can also perform some task that is thought to be difficult. For example, the Rabin encryption scheme [48] is said to be as hard as FACTORING (the problem of

determining prime numbers p and q given only their product pq). This means that if you have a device (*i.e.*, computer program, hardware, human being, etc.) that can discover secret messages created by the Rabin scheme, that device can easily be used to solve FACTORING. In particular, using the Rabin-breaking device to solve FACTORING would take little or no additional effort as compared to simply running the device to break the Rabin scheme.

What makes these relative security results useful is that the problems that they are based on are believed to be hard. Not only are these problems unsolved, but they have often been known and remained unsolved for decades (or even significantly longer). Significant cryptanalytic research goes into trying to solve these problems both in general and in specific cases (thereby making their hardness a useless assumption for security results). Research also goes into trying to show that the problems actually are hard.

In this dissertation, we examine three fundamental cryptographic assumptions. In Chapter 2, we examine an algorithm for solving DLOG, the discrete-logarithm problem, a problem whose difficulty the security of many encryption schemes relies upon. The security of the hierarchical identity-based encryption scheme described in Chapter 3 relies on the hardness of BDH, the bilinear-Diffie-Hellman problem. In Chapter 4, we rely on the hardness of RESIDUE $_r$, the r th residuosity problem.

1.3 Overview of Results

Chapter 2 focuses on algorithms for solving DLOG, the discrete-logarithm problem, which is defined to be, given a cyclic group generated by g and an element $y \in \langle g \rangle$, find $x \in \mathbb{Z}$ such that $y = g^x$. Because DLOG is thought to be hard to solve, the security of many cryptographic systems relies on DLOG's hardness. Put succinctly: were someone able to find a fast enough algorithm for solving DLOG, many cryptographic systems would become insecure.

Many cryptanalytic results focus on constructing algorithms for solving DLOG over particular types of groups (*e.g.*, \mathbb{Z}_p^\times , $\mathbb{F}_{2^m}^\times$, various elliptic curves). While subexponential algorithms for DLOG in certain groups have been found, there are still parameters

for cryptographic algorithms relying on the hardness of DLOG for which these DLOG algorithms remain prohibitively slow. If a fast enough algorithm for DLOG were found, it could be possible for cryptographers to change to a new group (*e.g.*, if the algorithm for DLOG only worked for groups of the form \mathbb{Z}_p^\times , the cryptographers could opt to use elliptic curves). This is what leads to the interest in “generic” DLOG algorithms, which do not rely upon any properties of the group over which the discrete logarithm is computed.

Shoup [52] showed that any algorithm for DLOG requires at least $\Omega(\sqrt{p})$ group operations (where p is the largest prime dividing the order of the group). Given such a bound, the natural complementary question is: can this bound be achieved? The baby-step giant-step algorithm achieves the bound, but it requires $\Theta(\sqrt{p})$ space. The other generic algorithm which claims to run in $O(\sqrt{p})$ steps is the Pollard rho algorithm. The Pollard rho algorithm requires $O(1)$ space, but it has a problem: it relies on a false assumption. It should be noted that, in practical applications, the Pollard rho algorithm appears to work perfectly; however, there is no formal guarantee. It is this lack that leads to the work in Chapter 2.

Chapter 2 is based on cursory work with Ramarathnam Venkatesan [32]. We focus on a new algorithm, based on the Pollard rho algorithm, which we call the Cayley rho algorithm. It utilizes random walks on random Cayley graphs to construct an algorithm for DLOG that runs in $\tilde{O}(\sqrt{p})$ time, uses $O(1)$ space, and does not rely on provably false assumptions. In addition to results for the Cayley rho algorithm, several intermediate results are of independent interest, as are several related results also presented in Chapter 2.

Chapter 3 discusses hierarchical identity-based encryption (HIBE), a hierarchical extension to identity-based encryption. Identity-based encryption is public-key encryption in which any string (*e.g.*, the recipient’s e-mail address) can be used as a public key (thus obviating the need for significant parts of public-key infrastructure). The initial realization of IBE schemes does not naturally support the hierarchical infrastructure standard to public-key encryption schemes. Extending an IBE to an HIBE has several practical advantages; the two main advantages examined in Chapter 3 are workload distribution and multi-level key escrow.

Though IBE (and HIBE) schemes no longer require public-key lookup, they do still require each user to request their own private key (only once). A basic IBE scheme requires one trusted key-distribution center to handle all of these requests, whereas an HIBE scheme allows a primary center to distribute its authority to subordinate centers. One example of a practical implementation of such a hierarchy would have a primary key-distribution center giving private keys to corporations, and those corporations giving private keys to their employees (with possibly additional levels of hierarchy imposed by the companies' organizational structures). The HIBE model also allows for escrow-related applications, including the natural consequence of the key-distribution structure: an HIBE scheme could be made to allow bosses the ability to read their subordinates' e-mail.

Identity-based encryption is a new and exciting cryptographic tool. Hierarchical IBE schemes are a natural and useful extension of IBE schemes, thus making them a prime candidate for study. Most of the results of Chapter 3 are based on earlier work with Ben Lynn [31].

Chapter 4 studies the application of the r th-order-residue symbol to cryptography. The key property of the r th-order-residue symbol is that it equals 1 when applied to a number of the form x^r (this concept is by no means degenerate; for arbitrary groups and arbitrary r , there is no reason to expect every element to have an r th root). While applications of the Jacobi symbol to cryptography have been extensively studied, its natural extension for $r > 2$, higher-order residuosity, remains largely ignored.

We present a public-key encryption scheme which works with r th residues modulo a reusable modulus $N = pq$. To encrypt a message, any party uses the publicly known N and r , as well as a publicly known $g \in \mathbb{Z}_N^\times$. As one would expect, any party who knows the factorization of N will be able to decrypt; however, this system has an additional property: any party that knows a nondegenerate root of unity μ can decrypt messages. Under the generally accepted assumptions that RESIDUE_r (the problem of distinguishing r th residues from arbitrary elements of \mathbb{Z}_N^\times) is hard and that N cannot easily be factored from knowing μ , we see that this creates three classes of users.

The separation into three types of users allows for several interesting properties.

First, the user(s) who know the factorization of N can change the list of users who can decrypt simply by changing r (and thus requiring a new μ) — there is no need to construct a new N . Other applications follow from allowing partial decryption: the scheme is constructed so that when given μ^d (for a d that divides s), a user is able to determine $m \bmod d$ from the encryption of m , thus allowing, for example, for various partial escrow applications.

We also present an additional application of the residue symbol: an extension of Peralta and Okamoto's improvement [44] to Lenstra's elliptic-curve method [36]. Peralta and Okamoto present an algorithm that outperforms Lenstra's algorithm on N of the form pq^2 by using Jacobi symbols. We show that their notions can be extended to work with pq^r for small $r > 2$. We also note that the only other algorithm for factoring numbers of the form pq^r [13] works only when r is relatively large (*i.e.*, $r \approx \sqrt{\log p}$).

Both applications give assorted uses for a new cryptologic tool, the r th-power-residue symbol, in various aspects of cryptology. This work is the result of joint work with Dan Boneh.

Chapter 2

Random Cayley Graphs and Discrete Logarithms

2.1 Introduction

The discrete logarithm problem (DLOG) defined over abelian groups plays a fundamental role in cryptography as a basis for many primitives (*e.g.*, Diffie-Hellman key exchange, DSS, and ElGamal signatures). The algorithms to solve DLOG fall into two types: the generic, black-box, exponential-time algorithms that use only the group structure (*e.g.*, baby-step giant-step and Pollard rho) and the domain-specific subexponential algorithms (*e.g.*, index calculus methods), which are not yet known to exist for groups over elliptic curves. Because of its generality and because it uses a very small amount of space, Pollard rho [46] is attractive (both practically and theoretically) and has inspired many works, including the parallelization by van Oorschot and Wiener [60] and by Pollard [47].

Surprisingly, there has never been a formal analysis of the classic Pollard rho without Pollard's heuristic assumptions. The standard analysis approximates the rho walk with a totally random walk (*i.e.*, a walk which at every step randomly and independently jumps to another group element) and then infers the existence (with high probability) of a cycle of length \sqrt{p} using the birthday paradox. But, in reality, the walk is far from random: the algorithm only makes a *deterministic walk* (which

is crucial for Floyd's algorithm to find a cycle using only a small amount of space) on a 3-regular directed graph over \mathbb{Z}_p^\times that is constructed randomly (from a subset of all possible such graphs). By using a random oracle for the moves to the *neighboring* nodes, Teske [56, 57] has analyzed both the original Pollard rho as well as more general k -regular graphs (for $k \geq 3$); for $k \geq 6$ she derives an $O(\sqrt{|G|})$ bound for finite abelian groups using a result of Hildebrand.

Earlier, Bach [5] studied Pollard rho for factoring and showed that the probability that a specific Pollard rho algorithm discovers a factor p of a given composite in k steps is (for fixed k) $\binom{k}{2}/p + O(p^{-3/2})$. Using this, he showed that the probability of success for the method is $c(\log^2 p)/p$ (for some $c > 0$), which is only slightly better than the obvious bound of $1/p$. Additionally, he makes some arithmetic conjectures (involving a form of pairwise independence for the number of points on some curves) that yield a satisfactory analysis of the rho method. He points out that these conjectures can be rigorously settled one way or the other, unlike the stochastic assumptions usually made. In fact, the rho method has always worked quite well in practice for the groups and sizes it has been tried on but, for all we know, we *might* have been implicitly relying on some properties specific to the underlying groups which yield results close to what the stochastic assumptions would predict; in principle, Pollard's assumptions may fail in a new untried group or size. It would be desirable to know if the rho method can be adapted for all abelian groups, thus confirming its expected universality.

To remove the need for these types of assumptions, we explicitly introduce randomness by modifying the algorithm slightly and by basing our treatment on random walks on Cayley graphs over abelian groups. Recall that an s -regular Cayley digraph (directed graph) on a group G has a set S of s generators. Its set of nodes is G and its edges are formed by connecting every α in G to αg_i (by a directed edge), for every $g_i \in S$. To solve for x given $y = g^x$ in G , we construct S with equal number of random powers of y and g . We start at a randomly chosen $z_0 \in G$ and move from z_i to z_{i+1} by multiplying z_i by a generator $g_{c(z_i, i)}$ look for a collision in the z_i s. (In fact, we will examine two methods for selecting c : randomly from functions of the form $c: G \rightarrow \{1, 2, \dots, s\}$ (in which case the generator is really $g_{c(z_i)}$) and randomly from functions of the form $c: \mathbb{N} \rightarrow \{1, 2, \dots, s\}$ (the generator is $g_{c(i)}$).

We show that, under a particular number-theoretic assumption, our modified algorithm, which is a *random walk with limited independence* on a *random* Cayley graph (*i.e.*, S is a random s -subset of G), solves DLOG in optimal time (up to logarithmic factors). These randomizations are essential and turn out to be just enough of a modification to allow for an analysis that works for all cyclic groups; thus, we are able to conclude that the rho method can be adapted for all abelian groups as expected. We note that a random choice of generators is important for two reasons: first, to show that the rho algorithm produces a nontrivial relationship (Theorem 2.2.4). Second, to guarantee that, over *any* abelian group, there exist Cayley graphs with underlying Markov chains that rapidly mix. (Without randomization, no such universal construction is known.) The rapid-mixing property is crucial for removing the dependence on a random-oracle assumption. Our result complements that of Shoup [52] who showed that *generic* algorithms for DLOG must take at least $\sqrt{|G|}$ steps. It would be interesting to know if random walks exploiting specific group properties yield faster algorithms.

As part of our analysis, we show that finding nontrivial cycles (*i.e.*, smaller than the group order) in random Cayley graphs over an abelian group G of order p is as hard as solving DLOG over G . These graphs are *succinctly presented* in the sense that they are defined by simple rules for moving from a node to its neighbors; they are, however, too huge to be explicitly stored. Our succinct graphs have girth (*i.e.*, the length of shortest cycle) $O(\log p)$; however, to computationally efficient algorithms, the girth appears to be exponential in $\log p$. One may view finding cycles in such a graph as finding two walks that start at the same node and collide later; this suggests some simple constructions for secure hash functions that are as secure as DLOG. Namely, we fix a random Cayley digraph with s generators. Assume that the input is given as x_1, x_2, \dots, x_t (for a suitable t), with $1 \leq x_i \leq s$. Assume that the graph has all its edges colored green. Now, for every $y \in G$, add directed red edges from y to y^2 . To compute the hash value, we start at some fixed h_0 , move along a green edge to h_1 by multiplying h_0 by the generator in S with index x_1 , follow the red edge to $h_2 = h_1^2$, and so on, alternating between crossing green edges and crossing red edges. The final endpoint is the hash value.

Lastly, we present implementation results supporting the theoretical results. We present run times for several different choices of $|S|$ and $p = |G|$ and see that the algorithm generally finds a discrete logarithm in $1.5\sqrt{p}$ to $2\sqrt{p}$ steps.

2.2 Preliminaries and Statement of Results

In this section we present relevant definitions, motivation, and statements of our results. Our study is from the point of view of path finding or navigating in exponentially large graphs that have simple rules for moving from one node to another. We assume that one is constrained to a limited amount of memory.

2.2.1 Cayley Digraphs

In view of the Pohlig-Hellman result on DLOG [45], we consider only prime-order groups; we denote the order of the group discussed in this paper by p . Such a group is cyclic (and, thus, abelian) with every element except the identity being a generator. For notions related to graph theory and random walks, we refer the reader to [12].

Let G be a multiplicative group of order p and let $S = \{g_1, g_2, \dots, g_s\}$ be a subset of G .

Definition 2.2.1 *The Cayley digraph generated by S is denoted by $\mathcal{G}(G, S) = (V, E)$ (we often simply write \mathcal{G}) and has the set of nodes $V = G$ and the set of (directed) edges $E = \{(g, gg_i) : g \in G, g_i \in S\}$.*

Most papers study undirected versions where, if $g \in S$, then $g^{-1} \in S$, and may additionally assume that the unit $1 \in S$ (i.e., all nodes have self loops); we cannot assume either of these conditions.

Definition 2.2.2 *A path of length t is a sequence (z_0, z_1, \dots, z_t) where, for every $i \in \{0, 1, \dots, t-1\}$, $(z_i, z_{i+1}) \in E$. A path is called a cycle if it also satisfies $z_t = z_0$.*

Since $g^p = 1$ for any $g \in G$, cycles occur in \mathcal{G} trivially; we will be interested only in *nontrivial* cycles (those having length $t \in [1, p-1]$). We assume that all

our paths and cycles are nontrivial and have length $t \leq \Lambda$ for a fixed constant $\Lambda = (\log^{O(1)} p)\sqrt{p} = o(p)$. Having $t = o(p)$ avoids wraparound problems even when we add the lengths of a constant number of paths.

In this paper, our main parameter is $s = O(\log p)$, where p is large enough to make DLOG hard; however, path lengths t can be exponentially large in s . Since G is abelian, paths (and cycles) of length t admit succinct representations of size $O(s \log t)$ as follows: given a path (or cycle), we write it as $\mathbf{x} = (x_1, x_2, \dots, x_s) \in \{0, 1, \dots, t\}^s$, where x_i is the number of the edges of the form (g, gg_i) in the path. This leads to our next definition:

Definition 2.2.3 *An s -tuple (x_1, x_2, \dots, x_s) is said to be the path type (or type) for a path (z_0, z_1, \dots, z_t) if, for $1 \leq i \leq s$, $x_i = |\{1 \leq j \leq t : z_j = g_i z_{j-1}\}|$.*

We will write $\|\mathbf{x}\|$ for the *length* of a path type $\mathbf{x} = (x_1, \dots, x_s)$. $\|\mathbf{x}\| := \sum_{i=1}^s x_i$; notice that the length of \mathbf{x} is the length (t) of any path whose path type is \mathbf{x} .

Definition 2.2.4 *For $\mathbf{x} \in \mathbb{Z}_p^s$, we define the vector greatest common divisor (vgcd) as $\text{vgcd}(\mathbf{x}) := \gcd(x_1, x_2, \dots, x_s)$.*

Now we come to the number-theoretic assumption upon which our algorithm relies:

Assumption 2.2.1 (VGCD) *For $\mathbf{x} \in \mathbb{Z}_p^s$, if $\|\mathbf{x}\| > \log p$, then $\Pr_c[\text{vgcd}(\mathbf{x}) > 1] < \frac{1}{8p}$.*

Succinct Graphs

We say that $\mathcal{G} = \mathcal{G}(G, S)$ is a *random Cayley digraph over G* if the elements of S are picked from G randomly and independently. By a *graph adjacency function* (for a graph (V, E)), we mean some algorithm to compute $f(u, i) = v$, where v is the i th ordered neighbor (under some predefined ordering) of the node u . If the graph is s -regular, then the edges can be colored, for example, with s colors (so, for any particular node, the d outgoing edges will all have different colors), and we set $f(u, i) = v$ if the edge (u, v) has the i th color. A graph is *succinctly presented* (or *succinct*) if there is a graph adjacency function $f(u, i)$ that runs in time $|u|^{O(1)}$, where

$|u|$ is the length of its label. We note that Cayley graphs over \mathbb{Z}_p^\times are succinct because one can take the standard binary representation of integers as the label and compute $f(\alpha, i)$ as αg_i , where g_i is the i th generator in S . Another example of a succinct graph is the k -dimensional hypercube with vertex set \mathbb{Z}_2^k with vertices connected exactly when they differ in one co-ordinate.

Definition 2.2.5 *A navigation algorithm is a function $h: G \times \mathbb{N} \rightarrow G$. (It is defined to construct a sequence, starting at z_0 , as $z_{i+1} := h(z_i, i)$.) When $h(z_i, i) = z_i g_{c(i)}$ (for all $i \in \mathbb{N}$) for some $c: \mathbb{N} \rightarrow \{1, 2, \dots, s\}$, we say that h is the navigation algorithm induced by c . The function c is called a generator selector.*

2.2.2 Limited Independence

Definition 2.2.6 *A sequence of random variables z_0, z_1, \dots, z_t is called m -wise independent if any subsequence of at most m variables is independent. A 2-wise-independent sequence is also called a pairwise-independent sequence. A function $f(x)$ is m -wise independent if, for any sequence of inputs $\{z_i\}_{i=0}^t$, the sequence $\{f(z_i)\}_{i=0}^t$ is m -wise independent.*

In our case, our m -wise independent variables will be uniformly distributed. We will randomly choose polynomials of degree $m - 1$ defined over an extension field of \mathbb{F}_2 — notice that such polynomials are m -wise independent. Indeed, given $\mathbf{z} = (z_0, z_1, \dots, z_{m-1})$ with distinct z_i and given $\mathbf{y} = (y_0, y_1, \dots, y_{m-1})$, one can find a polynomial with $f(z_i) = y_i$: set V to be the m by m matrix with entries $v_{ij} = z_i^j$ and solve the equation $\mathbf{y} = V\mathbf{f}$ (which is solvable since V is a Vandermonde (and, hence, invertible) matrix) for $\mathbf{f} = (f_0, f_1, \dots, f_{m-1})$ and set $f(x) := \sum_{i=0}^{m-1} f_i x^i$. We note that if we truncate each of the outputs of $f(z_i)$ (to some number of least-significant bits), we will still have an m -wise-independent sequence. To see this, note that in this case we are given only the truncated bits of entries in y and we may arbitrarily extend them to fully specify a vector y and proceed as before.

2.2.3 Finding Cycles in Succinct Graphs and DLOG

While finding paths and cycles efficiently in the usual graphs is well understood, finding paths and cycles in succinct graphs using only a small amount of space may be hard (though, in some cases, such as hypercubes, this is trivial).

Indeed, one may view the classic Pollard rho for solving $y = g^x$ as a method to

1. define (using y and g) a succinctly presented graph together with its navigation algorithm h and
2. find a cycle in the succinct graph and then solve a linear equation to solve DLOG.

Our modification to Pollard rho differs only in step 1 and is designed to bound, without using heuristic assumptions, the run time and the success probability in step 2.

Pollard Rho Algorithm

Let $g \neq 1$ be fixed. Given $y \in G = \langle g \rangle$, the task is to find x such that $y = g^x$. The algorithm (in some simple way) partitions G into three approximately equal-sized sets T_1 , T_2 , and T_3 (taking care that $1 \notin T_3$). Now, define the navigation algorithm $h_\rho: G \times \mathbb{N} \rightarrow G$ as:

$$h_\rho(z_i, i) = \begin{cases} z_i g & \text{for } z_i \in T_1 \\ z_i y & \text{for } z_i \in T_2 \\ z_i^2 & \text{for } z_i \in T_3 \end{cases} .$$

Starting with $z_0 = g^r$, construct a sequence $\{z_i\}_{i=0}^t$ with $z_{i+1} = h_\rho(z_i, i)$ until a collision occurs (*i.e.*, $z_u = z_v$ for some $u \neq v$). Given such a collision, x can likely be computed: let $z_u = z_0^{2^c} g^a y^b$ and $z_v = z_0^{2^{c'}} g^{a'} y^{b'}$, where $a, b, 2^c, a', b', 2^{c'} \in \mathbb{Z}_p$ are known from the iterative construction. Then $z_u = z_v$ is equivalent to $(b - b')x = (a' - a) + r(2^{c'} - 2^c) \pmod{p}$ (recall $p = |G|$). We can finish if the inverse $(b - b')^{-1}$ exists. To find a collision (a cycle) in $\{z_i\}_{i=0}^t$, one usually uses Floyd's algorithm, which needs only a small amount of space to keep track of construction of z_i and z_{2i} (for increasing i) until $z_i = z_{2i}$. If there is a cycle of length t , Floyd's algorithm takes $\Theta(t)$ steps.

Remark 2.2.1 *It is crucial that h above is deterministic if one wants to preserve the main advantages of using only a small amount of space and being able to avoid exhaustive search over the entire group. As noted earlier, in standard analysis for the rho method, one treats the z_i s as if they were random and independent (equivalently, one treats the graph as a complete graph (i.e., one with p^2 edges — an edge between every $v, v' \in V$) and the navigation function h as if it were chosen randomly from the set of all functions from G to G) and uses the birthday paradox to bound $t = O(\sqrt{p})$. Also, we note that in the standard analysis of Pollard rho, there is no formal guarantee that $(b-b')^{-1}$ exists with some specified probability (and $b-b'$ must be invertible mod p in order to finally discover x).*

Cayley Rho Algorithm

Fix a (cyclic) group G of order p and a generator $g \in G$ with respect to which we will solve DLOG. Where s is the size of $S \subseteq G$ (the set of generators for the Cayley graph), we, for convenience, assume that s is a power of 2. (Experiments show that when s is at least $4 \log_2 p$, the Cayley rho algorithm performs better than the Pollard rho; see Section 2.7.) We fix an extension field E/\mathbb{F}_2 with $[E : \mathbb{F}_2] = 3 \lceil \log p \rceil$ (unless otherwise stated, “log” always means the base-2 logarithm). Define C' to be the set of all degree- m (for an m that is polylogarithmic in p) polynomials from E to E . Let $y = g^x$ be given. We construct the Cayley algorithm CR-DLOG(y) (to find x) as follows:

1. **Defining the succinct graph:** Randomly choose $r_1, r_2, \dots, r_s \in \mathbb{Z}_p$. Then, use the r_i to construct $(g_1, g_2, \dots, g_s) \in G^s$: set (g_1, g_2, \dots, g_s) to be a random permutation of $(g^{r_1}, g^{r_2}, \dots, g^{r_{s/2}}, y^{r_{s/2+1}}, y^{r_{s/2+2}}, \dots, y^{r_s})$. Let $S := \{g_1, g_2, \dots, g_s\}$ and let $\mathcal{G} = \mathcal{G}(G, S)$ be the *random* Cayley graph generated by S over G . (For convenience, we assume, for all $i \neq j$, that $g_i \neq 1$ and $g_i \neq g_j^{-1}$ (see Remark 2.2.3).)

Initializing the navigation algorithm: We randomly choose and fix a polynomial $c' : E \rightarrow E$ from C' .

Computing $h(z_i, i)$: Given $i \in \{0, 1, \dots, \Lambda\}$ and $z_i \in G$, we define $z'_i \in E$

to be a standard $\lceil \log p \rceil$ -bit binary representation of z_i padded with a suitable prefix of zeros. Define $c_E: E \rightarrow \{1, 2, \dots, s\}$ so $c_E(z'_i)$ is the value represented by the $\log_2 s$ least-significant bits of the binary representation of $c'(z'_i)$. Finally, define $c(z_i) := c_E(z'_i)$. Then $h: G \times \mathbb{N} \rightarrow G$ is simply the navigation function induced by c : $h(z_i, i) = z_i g_{c(z_i)}$.

2. As in Pollard rho, we can use a procedure `FIND-CYCLE(\mathcal{G})` which outputs the type $\mathbf{x} = (x_1, x_2, \dots, x_s)$ representing a cycle in \mathcal{G} (*i.e.*, $\prod g_i^{x_i} = 1$). If the cycle is trivial, we repeat the entire algorithm; else we solve a linear equation (described below). We do this because, when the cycle is trivial, the equation cannot be solved (*i.e.*, the equation is $0x = 0$) and `CR-DLOG` must be restarted. (See Theorem 2.2.4 for further details.)

Remark 2.2.2 *Van Oorschot and Wiener [60] present a method for parallelizing collision-search algorithms that allows for a factor of k speedup by using k processors. Their method also applies to our algorithm; in other words, one can find discrete logarithms with the Cayley rho algorithm in $\tilde{O}(\sqrt{p}/k)$ time using k parallel processors.*

Remark 2.2.3 *We will assume that S is formed by picking s elements randomly and independently from G . These need not be distinct, so S can be a multiset. However, since we choose s to be polynomial in $\log p$ we expect (with high probability) that all the elements in S are unique. Even so, in Corollary 2.2.12 we show that any offset in our fundamental probability calculations induced by a small number of repeated elements is inconsequential.*

2.2.4 The Basic Walk and the Natural Walk

Pollard's original walk chooses the successive nodes by multiplying by a function of the current node; that is, $z_{i+1} := z_i f(z_i)$. In the walk that we actually implement — which we call the *basic walk* — we take steps similarly; $f(z_i) = g_{c(z_i)}$ for a randomly chosen c . However, we do most of our analysis (Section 2.3.3 is the exception) in terms of what we call the *natural walk*, where the successive nodes are chosen as a function of how many total steps have been made: $z_{i+1} := z_i g_{c(i)}$.

The most important realization about these two walks is that, until there is a collision, their distributions are indiscernible. Since our main result is bounding the probability that (after t steps) there have been no collisions, we can prove the result for either walk. We implement the basic walk so that Floyd's algorithm (looking for a collision of the form $z_i = z_{2i}$) will give us the advantage of constant space. We analyze the natural walk as it simplifies our calculations.

2.2.5 Notational Conventions

Conventions we use throughout the paper include denoting the path by z_0, z_1, \dots, z_t and, for the natural walk ($z_{i+1} = z_i g_{c(i)}$), writing C as the set of all generator selectors $c: \{0, 1, \dots, t-1\} \rightarrow \{1, 2, \dots, s\}$. Notice that the random walk is completely specified by c (and z_0).

Let $\Omega_t := \{\mathbf{x} \in \mathbb{Z}_p^s : \sum_{i=1}^s x_i = t\}$. Define a function $\psi_c: \mathcal{P}(\{0, 1, \dots, t-1\}) \rightarrow \Omega_t$ so, for $A \subseteq \{0, 1, \dots, t-1\}$, $\psi_c(A) = \mathbf{x} = (x_1, x_2, \dots, x_s)$, where (for each $1 \leq j \leq s$) $x_j := |c^{-1}(j) \cap A|$. In other words, the random walk induced by c picks each generator g_i a total x_i times during the $|A|$ (possibly nonconsecutive) steps (induced by A) of the random walk. So $\psi_c(A)$ is simply the type of the subpath induced by A . For notational convenience, we will write $\psi_c(t)$ for the special case $\psi_c(\{0, 1, \dots, t-1\})$, the type of the path made from the first t steps.

View the group S_s (of permutations of $\{1, 2, \dots, s\}$; not to be confused with the generators $S \subseteq G$) as acting on Ω_t and denote its orbits by T_1, T_2, \dots, T_N . We note that $\mathbf{x} = (x_1, x_2, \dots, x_s)$ and \mathbf{y} both belong to the same orbit T_j if and only if \mathbf{x} is a permutation of \mathbf{y} (*i.e.*, $\mathbf{y} = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(s)})$ for some $\sigma \in S_s$). Clearly, this induces an equivalence relation, and we write $\mathbf{x} \sim \mathbf{y}$ if and only if $\mathbf{x}, \mathbf{y} \in T_j$ for some j . As usual, we say that T_j is the orbit of \mathbf{x} . An important fact here is that if $\mathbf{x} \sim \mathbf{y}$, then $\Pr_{c \in C}[\psi_c(t) = \mathbf{x}] = \Pr_{c \in C}[\psi_c(t) = \mathbf{y}]$.

We define a function $\phi_S: \mathbb{Z}_p^s \rightarrow G$ as $\phi_S(\mathbf{x}) = \prod_{i=1}^s g_i^{x_i}$. Without loss of generality, we may assume that the starting point of our walk is the identity; as such, $\phi_S(\mathbf{x})$ is the endpoint of any walk whose type is \mathbf{x} (more generally, the endpoint z_t equals $z_0 \phi_S(\mathbf{x})$).

2.2.6 Our Results

While Theorem 2.2.2 is the most fundamental result of this paper, it should be noted that several of the ancillary results are of independent interest.

We note here that the main result (Theorem 2.2.2) uses Theorems 2.2.5 and 2.2.4.

Theorem 2.2.2 (Optimality of the Cayley Rho Algorithm) *Given an abelian group G of order p , CR-DLOG will expect to solve DLOG in no more than $(4 \log^3 p) \sqrt{p}$ steps.*

Proof. The theorem follows immediately from Theorems 2.2.5 and 2.2.4 (with $s = \log p$). □

Theorem 2.2.3 (Rapid Mixing (Basic Walk)) *Let \mathcal{G} be a random Cayley digraph over an abelian group G of prime order p and let $z_0 \in G$ be arbitrary. Starting from z_0 , construct a t -step (totally independent) random walk using the basic walk ($z_{i+1} = z_i g_{c(z_i)}$). If $t \geq 24 \log p$, then, for any $\alpha \in G$, $|\Pr_{S,c}[z_t = \alpha] - 1/p| < 3p^{-2}$.*

Proof. This proof is the content of Section 2.3.3. □

Theorem 2.2.4 (Degeneracy) *If the Cayley rho cycle-finding subroutine FIND-CYCLE returns successfully, CR-DLOG will solve DLOG with probability at least $2/s^2$; thus, the expected number of calls to FIND-CYCLE is at most $s^2/2$.*

Proof. This proof is the content of Section 2.4. □

Notice that, in essence, this theorem states that finding cycles in random Cayley graphs is at least as hard as solving DLOG.

Theorem 2.2.5 (Cycle-Finding Time) *Let CR-DLOG take $(8 \log p) \sqrt{p}$ moves on the graph. Then, under the VGCD assumption, the probability (over the random choices made by CR-DLOG) of a nontrivial cycle occurring is at least $\frac{1}{2}$.*

Proof. This proof is the content of Section 2.5. □

2.2.7 The Pairwise Independence of ϕ_S

Various theorems and lemmata will take advantage of the pairwise independence of ϕ_S , so we devote this section to discussing it.

We fix $g \neq 1$ (so that $G = \langle g \rangle$). We denote by $t_{\mathbf{x}} := \sum_i x_i$ the length of type \mathbf{x} .

Now we prove a simple lemma that helps characterize linearly dependent types:

Lemma 2.2.6 *Two types \mathbf{x} and \mathbf{y} are linearly dependent if and only if $t_{\mathbf{y}}\mathbf{x} = t_{\mathbf{x}}\mathbf{y}$.*

Proof. If $t_{\mathbf{y}}\mathbf{x} = t_{\mathbf{x}}\mathbf{y}$ and at least of \mathbf{x} , \mathbf{y} is nonzero — without loss of generality, $\mathbf{y} \neq \mathbf{0}$ — then $t_{\mathbf{y}} \neq 0$, so $t_{\mathbf{y}}\mathbf{x} + t_{\mathbf{x}}\mathbf{y} \neq 0$, proving that \mathbf{x} and \mathbf{y} are linearly dependent. In case $\mathbf{x} = \mathbf{y} = \mathbf{0}$, we have $1\mathbf{x} + 1\mathbf{y} = \mathbf{0}$ proving that \mathbf{x} and \mathbf{y} are linearly dependent.

To see the reverse implication, assume that \mathbf{x} and \mathbf{y} are linearly dependent. So we may choose $\alpha, \beta \in \mathbb{Z}_p$ (not both 0) such that $\alpha\mathbf{x} + \beta\mathbf{y} = \mathbf{0}$. This means that $\alpha t_{\mathbf{x}} + \beta t_{\mathbf{y}} = 0$, so $-\alpha t_{\mathbf{x}} = \beta t_{\mathbf{y}}$. Hence, $\alpha(t_{\mathbf{y}}\mathbf{x} - t_{\mathbf{x}}\mathbf{y}) = \alpha t_{\mathbf{y}}\mathbf{x} + \beta t_{\mathbf{y}}\mathbf{y}$, which equals $t_{\mathbf{y}}(\mathbf{0})$, so either $\alpha = 0$ or $t_{\mathbf{y}}\mathbf{x} - t_{\mathbf{x}}\mathbf{y} = \mathbf{0}$. In the latter case, we are done. If $\alpha = 0$, then $0\mathbf{x} + \beta\mathbf{y} = \mathbf{0}$ so, $\beta = 0$ or $\mathbf{y} = \mathbf{0}$. Since we cannot have both $\alpha = 0$ and $\beta = 0$, we know that $\mathbf{y} = \mathbf{0}$ (and, thus, that $t_{\mathbf{y}} = 0$). Hence, $t_{\mathbf{y}}\mathbf{x} = 0\mathbf{x} = \mathbf{0} = t_{\mathbf{x}}\mathbf{0} = t_{\mathbf{x}}\mathbf{y}$. \square

We present a corollary that will be used by Corollary 2.2.10:

Corollary 2.2.7 *Let $\mathbf{x} \neq \mathbf{y}$ be types. In both of the following cases, \mathbf{x} and \mathbf{y} are linearly independent:*

- (a) $t_{\mathbf{x}} = t_{\mathbf{y}}$ or
- (b) \mathbf{x} and \mathbf{y} and both are binary vectors (neither all zeroes).

Proof. (a) Since $\mathbf{x} \neq \mathbf{y}$, at least one of \mathbf{x} , \mathbf{y} is not $\mathbf{0}$; we conclude that $t_{\mathbf{x}} = t_{\mathbf{y}} > 0$. Thus, $t_{\mathbf{y}}\mathbf{x} \neq t_{\mathbf{y}}\mathbf{y}$. Since $t_{\mathbf{y}} = t_{\mathbf{x}}$, we know that $t_{\mathbf{y}}\mathbf{y} = t_{\mathbf{x}}\mathbf{y}$, and, thus, $t_{\mathbf{y}}\mathbf{x} \neq t_{\mathbf{x}}\mathbf{y}$. Hence, Lemma 2.2.6 tells us that \mathbf{x} and \mathbf{y} are linearly independent.

(b) Since $\mathbf{x} \neq \mathbf{y}$, we may choose an $i \in \{1, 2, \dots, s\}$ such that $x_i \neq y_i$. Without loss of generality, we may assume that $x_i = 1$ and $y_i = 0$. Since $\mathbf{y} \neq \mathbf{0}$, $t_{\mathbf{y}} > 0$, so $t_{\mathbf{y}}x_i > 0 = t_{\mathbf{x}}y_i$. Thus, $t_{\mathbf{y}}\mathbf{x} \neq t_{\mathbf{x}}\mathbf{y}$ and we may again apply Lemma 2.2.6 to see that \mathbf{x} and \mathbf{y} are linearly independent. \square

Linear independence plays a key role in this section's main lemma (Lemma 2.2.9); it considers pairs of linearly independent paths and shows that randomly choosing the set S makes ϕ_S a pairwise-independent function on S . (Recall that $\phi_S(\mathbf{x}) = \prod_i g_i^{x_i}$.)

Before proving this section's main lemma, we show that, as a function of S (for a fixed \mathbf{x}), $\phi_S(\mathbf{x})$ distributes its output uniformly over \mathbb{Z}_p :

Lemma 2.2.8 *For any nonzero path type \mathbf{x} and any $\alpha \in G$,*

$$\Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = \alpha] = \frac{1}{p} .$$

Proof. Since $G = \langle g \rangle$, we may choose $a \in \mathbb{Z}_p$ such that $g^a = \alpha$. Define $\phi^*: \mathcal{S} \rightarrow G$ (for our fixed \mathbf{x}) as $\phi^*(S) := \phi_S(\mathbf{x})$. Notice that $\Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = g^a] = |\phi^{*-1}(g^a)| / |\mathcal{S}|$. Now we show that, for any $b, c \in \mathbb{Z}_p$, $|\phi^{*-1}(g^b)| = |\phi^{*-1}(g^c)|$.

Let arbitrary $b, c \in \mathbb{Z}_p$ be given and we will construct a function $f: \phi^{*-1}(g^b) \rightarrow \phi^{*-1}(g^c)$ that is injective. Denote by i the minimum value such that $x_i > 0$ (since $\mathbf{x} \neq 0$, such an i exists). If $S = (g_1, g_2, \dots, g_s)$, we define $f(S) := S' = (g'_1, g'_2, \dots, g'_s)$, where $g'_j = g_j$ for $j \neq i$ and $g'_i = g^{(c-b)x_i^{-1}} g_i$. Notice that if $\phi^*(S) = g^b$, then $\phi^*(S') = g^c$. Also notice that f is injective. Thus, $|\phi^{*-1}(g^b)| \leq |\phi^{*-1}(g^c)|$. As b and c were arbitrary, we can switch them to conclude the reverse inequality and, thus, the sought equality.

Set K to be the constant value equal to the size of any preimage $|\phi^{*-1}(g^b)|$. Then,

$$|\mathcal{S}| = \left| \bigcup_{a \in \mathbb{Z}_p} \phi^{*-1}(g^a) \right| = \sum_{a \in \mathbb{Z}_p} |\phi^{*-1}(g^a)| = pK ,$$

so $K = |\mathcal{S}|/p$. But $\Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = \alpha] = |\phi^{*-1}(g^a)| / |\mathcal{S}| = K / |\mathcal{S}|$. Hence, we conclude that $\Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = \alpha] = 1/p$. \square

Lemma 2.2.9 (Pairwise Independence) *If \mathbf{x} and \mathbf{y} are linearly independent types of lengths at most Λ , then ϕ_S is a pairwise-independent mapping, i.e., for any $\alpha, \beta \in G$,*

$$\Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = \alpha \wedge \phi_S(\mathbf{y}) = \beta] = \frac{1}{p^2} = \Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = \alpha] \Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{y}) = \beta] .$$

Remark 2.2.4 *The restriction of \mathbf{x} and \mathbf{y} to be linearly independent is essential. For example, if a length- t path with type \mathbf{x} ends at α , then one can “predict,” without knowing any other information about S , at which node all paths with type $\mathbf{y} := 2\mathbf{x}$ (of length $2t$) will end: α^2 .*

Proof. Since $G = \langle g \rangle$, we may choose $a, b \in \mathbb{Z}_p$ such that $g^a = \alpha$ and $g^b = \beta$. Let the number of nonzero entries in the vector $\mathbf{x} = (x_1, x_2, \dots, x_s)$ be r . (Notice that, since \mathbf{x} and \mathbf{y} are linearly independent, $r > 0$.) We will assume that, by reordering indices if necessary, $x_i \neq 0$ for $1 \leq i \leq r$ and $x_i = 0$ for $r < i \leq s$. Define $\boldsymbol{\delta} := (\delta_1, \delta_2, \dots, \delta_s)$ so that $\mathbf{y} = \mathbf{x} + \boldsymbol{\delta}$. Put $t := \|\mathbf{x}\| = \sum_{i=1}^s x_i$ and $d := \|\boldsymbol{\delta}\| = \sum_{i=1}^s \delta_i$. We have $t \not\equiv 0 \pmod{p}$ since $t > 0$ ($t \neq 0$ because \mathbf{x} and \mathbf{y} are linearly independent) and $t \leq \Lambda < p$. Define $a, b, \gamma_1, \gamma_2, \dots, \gamma_s \in \mathbb{Z}_p$ so $g^a = \alpha$, $g^b = \beta$, $g^{\gamma_i} = g_i$ (for $1 \leq i \leq s$) and notice that the conditions $\phi_S(\mathbf{x}) = \alpha$ and $\phi_S(\mathbf{y}) = \beta$ are equivalent to

$$\sum_{i=1}^r \gamma_i x_i = a \pmod{p} \quad \text{and} \quad \sum_{i=1}^s \gamma_i \delta_i = b - a \pmod{p}. \quad (2.1)$$

We now consider two cases:

Case 1 ($\delta_u \neq 0$ for some $u \in [r+1, s]$): From the definition of r , we have that $x_i \neq 0$ for all $i \leq r$; hence, $x_r \neq 0$. Then, for each of the $|G|^{s-2}$ choices for γ_i with $i \in \{1, 2, \dots, s\} \setminus \{r, u\}$, we have one choice for (γ_r, γ_u) :

$$\gamma_r = x_r^{-1} \left(a - \sum_{i=1}^{r-1} \gamma_i x_i \right) \pmod{p} \quad \text{and} \quad \gamma_u = \delta_u^{-1} \left(b - a - \sum_{\substack{i=1 \\ i \neq u}}^s \gamma_i \delta_i \right).$$

This shows that $\Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = g^a \wedge \phi_S(\mathbf{y}) = g^b] = |G|^{s-2}/|G|^s = 1/p^2$. Lemma 2.2.8 then shows that this equals $\Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = \alpha] \Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{y}) = \beta]$.

Case 2 ($\delta_i = 0$ for all $i > r$): We now introduce new variables λ_i with $\lambda_i := \gamma_i - \gamma_r$ for $1 \leq i < r$ and $\lambda_i = \gamma_i$ for $r \leq i \leq s$ (so each element of \mathcal{S} corresponds to a $(\lambda_1, \lambda_2, \dots, \lambda_s) \in G^s$) and, from the equations in 2.1, see that

$$\sum_{i=1}^{r-1} \lambda_i x_i + t \lambda_r = a \pmod{p} \quad \text{and} \quad \sum_{i=1}^{r-1} \lambda_i \delta_i + d \lambda_r = b - a \pmod{p}. \quad (2.2)$$

First, we show that $r > 1$. Since $\delta_i = 0$ for all $i > r$ (the definition of **case 2**), we conclude that $y_i = 0$ for all $i > r$. If $r = 1$, then $y_1 \mathbf{x} = x_1 \mathbf{y}$, contradicting that \mathbf{x} and \mathbf{y} are linearly independent. Hence, $r > 1$.

Notice that if $t\delta_i = dx_i$ for all $i \in \{1, 2, \dots, r\}$, we have that $t\boldsymbol{\delta} = d\mathbf{x}$ and, therefore, that $t\mathbf{y} = (d+t)\mathbf{x}$. Since this cannot happen (as \mathbf{x} and \mathbf{y} are linearly independent), we conclude that we may choose an $i^* \in \{1, 2, \dots, r\}$, such that $t\delta_{i^*} \neq dx_{i^*}$. Without loss of generality, $i^* < r$. Now, notice that for each of the $|G|^{s-2}$ choices for λ_i with $i \in \{1, 2, \dots, s\} \setminus \{i^*, r\}$, we have one choice for $(\lambda_{i^*}, \lambda_r)$, as seen by

$$\lambda_{i^*} = (\delta_{i^*} - dt^{-1}x_{i^*})^{-1} \left(b - a - dt^{-1}a - \sum_{\substack{i=1 \\ i \neq i^*}}^{r-1} \lambda_i (\delta_i - dt^{-1}x_i) \right) \pmod{p}$$

and

$$\lambda_r = t^{-1} \left(a - \sum_{i=1}^{r-1} \lambda_i x_i \right) \pmod{p}.$$

The proof ends as in **case 1**. □

We complete this section with three corollaries of Lemma 2.2.9:

Corollary 2.2.10 (a) On any $A \subseteq \Omega_t$ ($t \leq \Lambda$), the mapping ϕ_S is pairwise independent. (b) On $B := \{(x_1, x_2, \dots, x_s) : x_i \in \{0, 1\}, \text{ not all zero}\}$, ϕ_S is a pairwise-independent map. In this case, ϕ_S is a subset-product map on nonempty sets of generators.

Proof. (a) If $|A| < 2$, the statement is vacuously true. Otherwise, let $\mathbf{x} \neq \mathbf{y} \in A$ be given and notice that $t_{\mathbf{x}} = t = t_{\mathbf{y}}$, so Corollary 2.2.7(a) tells us that \mathbf{x} and \mathbf{y} are linearly independent, allowing us to apply Lemma 2.2.9.

(b) This follows immediately from Corollary 2.2.7(b) and Lemma 2.2.9. □

Corollary 2.2.11 If $\alpha \in G$, $A \subseteq \Omega_t$, and, for $S \in \mathcal{S}$, $N_S := |\phi_S^{-1}(\alpha) \cap A|$ (the number of types in A that ϕ_S maps to α), then

$$\mathbb{E}_{S \in \mathcal{S}}[N_S] = \frac{|A|}{p} \quad \text{and} \quad \text{Var}_{S \in \mathcal{S}}[N_S] = \frac{|A|}{p} \left(1 - \frac{1}{p} \right).$$

Proof. Notice that $N_S = \sum_{\mathbf{x} \in A} \chi_{\phi_S(\mathbf{x})=\alpha}$, so

$$\mathbb{E}_{S \in \mathcal{S}}[N_S] = \sum_{S \in \mathcal{S}} \frac{1}{|\mathcal{S}|} N_S = \sum_{\mathbf{x} \in A} \Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = \alpha] = |A| \cdot \frac{1}{p},$$

by Lemma 2.2.8. Since $\text{Var}_{S \in \mathcal{S}}[N_S] = \mathbb{E}_{S \in \mathcal{S}}[N_S^2] - (\mathbb{E}_{S \in \mathcal{S}}[N_S])^2$, we compute $\mathbb{E}_{S \in \mathcal{S}}[N_S^2]$:

$$\begin{aligned} \mathbb{E}_{S \in \mathcal{S}}[N_S^2] &= \sum_{S \in \mathcal{S}} \frac{1}{|\mathcal{S}|} \left(\sum_{\mathbf{x} \in A} \chi_{\phi_S(\mathbf{x})=\alpha} \right)^2 = \sum_{S \in \mathcal{S}} \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in A} \chi_{\phi_S(\mathbf{x})=\alpha} \chi_{\phi_S(\mathbf{y})=\alpha} \\ &= \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in A} \Pr_{S \in \mathcal{S}}[(\phi_S(\mathbf{x}) = \alpha) \wedge (\phi_S(\mathbf{y}) = \alpha)]. \end{aligned}$$

Corollary 2.2.7 tells us that \mathbf{x} and \mathbf{y} are linearly independent when $\mathbf{x} \neq \mathbf{y}$. When $\mathbf{x} = \mathbf{y}$, $\Pr_{S \in \mathcal{S}}[(\phi_S(\mathbf{x}) = 1) \wedge (\phi_S(\mathbf{y}) = 1)] = \Pr_{S \in \mathcal{S}}[(\phi_S(\mathbf{x}) = 1)]$, so, from Lemmata 2.2.9 and 2.2.8, we realize

$$\mathbb{E}_{S \in \mathcal{S}}[N_S^2] = |A|(|A| - 1) \frac{1}{p^2} + |A| \frac{1}{p}.$$

Thus, $\text{Var}_{S \in \mathcal{S}}[N_S] = \mathbb{E}_{S \in \mathcal{S}}[N_S^2] - (\mathbb{E}_{S \in \mathcal{S}}[N_S])^2 = |A| \frac{1}{p} \left(1 - \frac{1}{p}\right)$. \square

Corollary 2.2.12 *If $\mathcal{S}_0 \subseteq \mathcal{S}$ is such that $|\mathcal{S}_0| \geq (1 - \varepsilon/2)|\mathcal{S}|$ (with $\varepsilon \leq 1$), then, for any type \mathbf{x} (of length at most Λ) and $\alpha \in G$,*

$$\left| \Pr_{S \in \mathcal{S}_0}[\phi_S(\mathbf{x}) = \alpha] - \Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = \alpha] \right| \leq \varepsilon.$$

Proof. Notice that Lemma 2.2.8 tells us that $\Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = \alpha] = 1/p$.

Now realize that

$$\begin{aligned} \Pr_{S \in \mathcal{S}_0}[\phi_S(\mathbf{x}) = \alpha] &= \frac{\Pr_{S \in \mathcal{S}}[(\phi_S(\mathbf{x}) = \alpha) \wedge (S \in \mathcal{S}_0)]}{\Pr_{S \in \mathcal{S}}[S \in \mathcal{S}_0]} \leq \frac{\Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = \alpha]}{\Pr_{S \in \mathcal{S}}[S \in \mathcal{S}_0]} \\ &\leq \Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = \alpha] \frac{1}{1 - \frac{\varepsilon}{2}} = \frac{1}{p} \cdot \frac{1}{1 - \frac{\varepsilon}{2}}, \end{aligned}$$

the last step following from Lemma 2.2.8.

As $\varepsilon \leq 1$, we know that $\frac{1}{1-\frac{\varepsilon}{2}} \leq 1 + \varepsilon$, so

$$\Pr_{S \in \mathcal{S}_0} [\phi_S(\mathbf{x}) = \alpha] - \Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = \alpha] = \Pr_{S \in \mathcal{S}_0} [\phi_S(\mathbf{x}) = \alpha] - \frac{1}{p} \leq \frac{1}{p}(1 + \varepsilon) - \frac{1}{p} = \varepsilon \frac{1}{p} < \varepsilon .$$

To see the lower bound, define $\mathcal{S}_1 := \{S \in \mathcal{S} : \phi_S(\mathbf{x}) = \alpha\}$ (which is dependent upon \mathbf{x} and α) and notice that $|\mathcal{S}_1|/|\mathcal{S}| = \Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = \alpha] = 1/p$. Now notice that $|\mathcal{S}_1 \cap \mathcal{S}_0| = |\mathcal{S}_1| - |\mathcal{S}_1 \cap \overline{\mathcal{S}_0}| \geq |\mathcal{S}_1| - |\overline{\mathcal{S}_0}| \geq |\mathcal{S}_1| - (\varepsilon/2)|\mathcal{S}| = |\mathcal{S}|(1/p - \varepsilon/2)$. Hence,

$$\begin{aligned} \Pr_{S \in \mathcal{S}_0} [\phi_S(\mathbf{x}) = \alpha] - \Pr_{S \in \mathcal{S}} [\phi_S(\mathbf{x}) = \alpha] &= \Pr_{S \in \mathcal{S}_0} [\phi_S(\mathbf{x}) = \alpha] - \frac{1}{p} = \frac{|\mathcal{S}_1 \cap \mathcal{S}_0|}{|\mathcal{S}_0|} - \frac{1}{p} \\ &\geq \frac{|\mathcal{S}|}{|\mathcal{S}_0|} \left(\frac{1}{p} - \frac{\varepsilon}{2} \right) - \frac{1}{p} \\ &= \frac{1}{p} \left(\frac{|\mathcal{S}| - |\mathcal{S}_0|}{|\mathcal{S}_0|} \right) - \frac{|\mathcal{S}|}{|\mathcal{S}_0|} \left(\frac{\varepsilon}{2} \right) \geq -\frac{|\mathcal{S}|}{|\mathcal{S}_0|} \left(\frac{\varepsilon}{2} \right) \\ &\geq -\frac{1}{1-\frac{\varepsilon}{2}} \left(\frac{\varepsilon}{2} \right) \geq -\varepsilon . \end{aligned}$$

□

2.3 The Basic Walk

Outside of this section, we will examine the natural walk, the walk we get from selecting $z_0 \in G$ randomly, $c: \mathbb{N} \rightarrow \{1, 2, \dots, s\}$ randomly (from \mathcal{C}), and setting $z_{i+1} := h(z_i, i)$, the navigation algorithm induced by c (recall that we defined $h(z_i, i) = z_i g_{c(i)}$). However, in this section, we give an alternative walk and several results of independent interest, from a Markov chain perspective. Our alternative walk will use the navigation algorithm $h(z_i, i) = z_i g_{c(z_i)}$, which we note is equivalent to defining $z_{i+1} := h(z_i)$, where $h: G \rightarrow G$ is defined to be $\alpha \mapsto \alpha g_{c(\alpha)}$ for a $c: G \rightarrow \{1, 2, \dots, s\}$.

An initial direction to turn for proving a result like Theorem 2.2.3 is toward Markov-chain methods. Rapid mixing of Cayley graphs is well studied; however, we could not find a reference for the case of Cayley digraphs with both $O(\log p)$ generators and no self loops which states the required bound ($O(p^{-2})$ rather than $O(1)$) on the deviation from the uniform. It is simple to show, using elementary

matrix methods, the following: starting at an arbitrary z_i , if a purely random walk on an expander converges to an almost-uniform distribution $\mu: G \rightarrow [0, 1]$ in τ steps (*i.e.*, the node $z_{i+\tau}$ is almost-uniformly distributed), then, for any $t > \tau$, the distribution of z_{i+t} remains almost-uniformly distributed. These matrix methods do not easily transform to show that the distribution of z_{i+t} is almost-uniformly distributed when the walk steps are correlated.

This section focuses on results which are not necessary for proving the main theorem of the paper (Theorem 2.2.2), but are of interest in their own right. The section begins with a discussion of previous work (Section 2.3.1) for graphs similar to our graphs (but not exactly the same as our graphs; hence, we must come up with new results). We finish the section with a discussion of the Markov chain induced by the walk $\{z_i\}$ (it tends to a uniform stationary distribution (Section 2.3.2)).

2.3.1 Previous Work

We recall some standard definitions.

Definition 2.3.1 *The boundary of a $D \subseteq V$ is prescribed to be the set*

$$\partial D = \{v \in V : v \notin D \text{ and } v \text{ has incoming edge from some node in } D\} .$$

Definition 2.3.2 *If $U \subseteq V$ and for every subset W of U we have $|\partial W| \geq \varepsilon|W|$, then U is then called ε -expanding. We call the subgraph induced by an ε -expanding subset an ε -expanding graph. The entire graph $\mathcal{G} = (V, E)$ is called an ε -expander if every subset of size at most $\frac{|V|}{2}$ is ε -expanding.*

Normally, ε is taken to be a constant as the size of \mathcal{G} grows; one shows that on such expanders a random walk rapidly mixes in the sense that it reaches a distribution exceptionally close to its stationary (uniform) distribution in $O(\log p)$ steps. Cayley graphs and general expanders are the subject of extensive literature and the reader may consult the papers we cite for further references; our survey here is rather limited in scope.

Using the methods of Broder and Shamir [17], Alon and Roichman [2] have shown that the symmetric Cayley graphs with random $S \cup S^{-1}$ as generators are indeed expander graphs with overwhelming probability. They have also shown that these graphs cannot be expanders unless $s \geq k \log p$ for some constant k , a result which extends to our digraphs. This implies that if s is a constant (as in Pollard's algorithm, where $s = 3$), the expansion parameter ε cannot be a constant, and the mixing may be slower.

Similar methods are used by Roichman [49] to study both the directed and undirected cases for $s = \log^k p$, $k > 1$; for $k = 1$, Dou and Hildebrand [25] use representation theory for analysis. Directed expanders are analyzed by Mihail [39] and Fill [27]. Most of the literature deals with the case when the graph is either undirected or has self loops. The notions of expansion and rapid mixing are equivalent for undirected graphs (see [61]), but their relationship is unclear in the case of digraphs. We can show, using a paper of Babai [4], that the Cayley graphs are $\frac{1}{s^2}$ -expanders and, if self loops are allowed, we can bound the mixing time using the work of Fill [27]. In this section we give a direct analysis that yields required bounds of mixing for Cayley digraphs that do not have self loops. The walks in this section are purely random.

2.3.2 The Markov Chain Induced by G

We define our random walk on \mathcal{G} as follows: starting at an initial node z_0 , one picks, uniformly at random, one of the outgoing edges (say, $(z_0, z_0 g_i)$) and moves to the opposite node (*i.e.*, $z_1 := z_0 g_i$). Then we iterate this step, using independent coin flips at each node. The induced Markov chain (which we denote by **MC**) has the transition matrix M with entries $m_{\alpha\beta} = 1/s$ if there is an edge from the node α to node β (else it is zero); the adjacency matrix $A(\mathcal{G})$ has entries $a_{\alpha\beta} = s m_{\alpha\beta}$. Our graphs are directed and we need to prove many of their properties from scratch.

In the standard uses of Markov chains in algorithms, the aperiodicity (*i.e.*, the property that the gcd of all lengths of paths from any node to itself) is often achieved trivially by adding self loops to the nodes. This is not a viable option for us, as we need our graph to have a large girth (*i.e.*, the length of the shortest cycle).

Also, our graphs are strictly directed so, in addition to M and $A(\mathcal{G})$ having a main diagonal of zeroes, they are both asymmetric; if they were symmetric, present literature contains ample analysis (using eigenvalues) of the *mixing time* (*i.e.*, how fast the Markov chains reach their stationary distributions).

Since existing matrix-theory literature does not serve us, we will use probabilistic arguments to achieve the estimate we need for the deviation from the stationary (in this case, uniform) distribution. We also show (in this case) that if the purely random walk is convergent, then so is the related limited-independence random walk.

Here, we prove a fundamental result about **MC**:

Theorem 2.3.1 *For all but a negligible fraction of choices of S , **MC** has a stationary distribution. In addition, that distribution is the unique stationary distribution and it is the uniform distribution.*

Proof. Notice that (unless $S = \{1\}$) the elements of S generate G and, thus, the Cayley digraph \mathcal{G} is strongly connected (*i.e.*, **MC** is irreducible). For any irreducible Markov chain, by the Perron-Frobenius theorem (see, *e.g.*, [50]), the adjacency matrix has 1 as the maximal eigenvalue; additionally, this eigenvalue has multiplicity one. To guarantee a stationary distribution of the chain, we need only show that the chain is also aperiodic (which we do, for all but a negligible fraction of S in Lemma 2.3.2).

Note that the group structure imposes that the in-degree and the out-degree of any node are the same (both equal to $|S|$), making M doubly stochastic (*i.e.*, every column sums to one, as does every row). Hence, since **MC** has a stationary distribution, it must be the uniform distribution. \square

Lemma 2.3.2 *If $s > \log(4p/\delta)$ and $t > s$, then **MC** is aperiodic for all but less than a δ fraction of S .*

Proof. For any given S , a path of length t with an associated type $\mathbf{x} \in \Omega_t$ returns to its starting point if and only if $\phi_S(\mathbf{x}) = 1$. Now we bound from below the probability of the existence of paths of length t .

Let a be the number of types of length t and let the random variable $N_S := |\phi_S^{-1}(1) \cap \Omega_t|$ denote the number of paths of length t that ϕ_S maps to the identity.

Clearly, $a = \binom{t+s-1}{s-1}$. Since Corollary 2.2.11 tells us that $\mu := \mathbb{E}_{S \in \mathcal{S}}[N_S] = \frac{a}{p}$ and $\text{Var}_{S \in \mathcal{S}}[N_S] = a \frac{1}{p} \left(1 - \frac{1}{p}\right)$, we know that $\Pr_{S \in \mathcal{S}}[N_S = 0] \leq \Pr_{S \in \mathcal{S}}[|Y - \mu| \geq \mu] \leq \frac{\text{Var}_{S \in \mathcal{S}}[N_S]}{\mu^2} < \frac{p}{a}$.

Since $t > s$, we know that $\frac{t}{s-1} \geq 1$, so

$$a = \binom{t+s-1}{s-1} \geq \left(\frac{t+s-1}{s-1}\right)^{s-1} = \left(1 + \frac{t}{s-1}\right)^{s-1} \geq 2^{s-1} > \frac{2p}{\delta}.$$

Thus, the probability of the nonexistence of a path of length t (i.e., $\Pr_{S \in \mathcal{S}}[N_S = 0]$) is less than $\delta/2$.

Now notice that, since we also know that $\frac{t+1}{s-1} > 1$ and $t+1 > \log(2p/\delta)$, a similar argument shows that the probability of the nonexistence of a path of length $t+1$ is less than $\delta/2$.

Hence, the probability (of the two dependent events) that a path of length t and a path of length $t+1$ do exist is more than $1 - (\delta/2 + \delta/2) = 1 - \delta$. When paths of both lengths exist, aperiodicity follows since $\gcd(t, t+1) = 1$. \square

Remark 2.3.1 *The techniques used in Lemma 2.3.2 can also be used to conclude that, for almost every graph, the diameter of the graph is $\Theta(\log |G|)$ and the girth (i.e., the length of the shortest cycle) is $\Theta(\log |G|)$.*

2.3.3 Rapid Mixing (Basic Walk, Total Independence)

Here we present the mixing result for the basic walk:

Theorem 2.2.3 (Rapid Mixing (Basic Walk)) *Let \mathcal{G} be a random Cayley digraph over an abelian group G of prime order p and let $z_0 \in G$ be arbitrary. Starting from z_0 , construct a t -step (totally independent) random walk using the basic walk ($z_{i+1} = z_i g_{c(z_i)}$). If $t \geq 24 \log p$, then, for any $\alpha \in G$, $|\Pr_{S,c}[z_t = \alpha] - 1/p| < 3p^{-2}$.*

Proof. Set $W := \{1, 2, \dots, t\}$ and $\beta := \alpha z_0^{-1}$ and notice that, since $z_t = \alpha$ exactly when $\phi_S(\psi_c(W)) = \beta$, we may apply Theorem 2.3.3 to complete this proof. \square

Finally, we give the main result upon which Theorem 2.2.3 relies:

Theorem 2.3.3 *Let \mathcal{G} be a random Cayley digraph over an abelian group G of prime order p and let $z_0 \in G$ be arbitrary. Let $W \subseteq \{0, 1, \dots, \Lambda\}$. If $|W| \geq 24 \log p$, then, for any $\beta \in G$, $|\Pr_{S,c}[\phi_S(\psi_c(W)) = \beta] - 1/p| \leq 3p^{-2}$.*

Proof. In Section 2.2.5, we wrote T_1, T_2, \dots, T_N as the orbits of S_s acting on Ω_t . Without loss of generality, we may assume that $a < b \Rightarrow |T_a| \leq |T_b|$. Pick the smallest L such that $|T_L| \geq p^5$.

Fix $j \geq L$ and notice that, according to Lemma 2.3.5 (with $\delta = p^{-2}$ and $\varepsilon = p^{-2}$), $\Pr_{\mathbf{x}}[\phi_S(\mathbf{x}) = \alpha | \mathbf{x} \in T_j] \leq \frac{1}{p} + p^{-2}$, for all but a p^{-2} fraction of $S \in \mathbf{S}$ (which we will call \mathbf{S}_{GOOD}). Hence,

$$\begin{aligned} \Pr_{S,\mathbf{x}}[\phi_S(\mathbf{x}) = \alpha | \mathbf{x} \in T_j] &= \frac{1}{|\mathbf{S}|} \sum_{S \in \mathbf{S}} \Pr_{\mathbf{x}}[\phi_S(\mathbf{x}) = \alpha | \mathbf{x} \in T_j] \\ &\leq \frac{1}{|\mathbf{S}|} \left(\left(\frac{1}{p} + p^{-2} \right) |\mathbf{S}_{\text{GOOD}}| + 1 |\mathbf{S} \setminus \mathbf{S}_{\text{GOOD}}| \right), \end{aligned}$$

which is clearly no more than $\frac{1}{p} + 2p^{-2} - p^{-3} - p^{-4}$.

Since the T_j are disjoint, we know that (for all $S \in \mathbf{S}$) $\sum_{j=1}^{L-1} \Pr_{c \in \mathbf{C}}[\psi_c(W) \in T_j] = \Pr_{c \in \mathbf{C}}[\psi_c(W) \in \bigcup_{j=1}^{L-1} T_j]$. Since Lemma 2.3.6 tells us exactly that the latter is at most p^{-2} , we conclude that

$$\sum_{j=1}^{L-1} \Pr_{S,c}[\psi_c(W) \in T_j] = \frac{1}{|\mathbf{S}|} \sum_{S \in \mathbf{S}} \sum_{j=1}^{L-1} \Pr_{c \in \mathbf{C}}[\psi_c(W) \in T_j] \leq \frac{1}{|\mathbf{S}|} \sum_{S \in \mathbf{S}} p^{-2} = p^{-2}.$$

So we use these inequalities ($\Pr_{S,\mathbf{x}}[\phi_S(\mathbf{x}) = \alpha | \mathbf{x} \in T_j] \leq \frac{1}{p} + 2p^{-2} - p^{-3} - p^{-4}$ and

$\sum_{j=1}^{L-1} \Pr_{S,c}[\psi_c(W) \in T_j] \leq p^{-2}$ to realize that

$$\begin{aligned}
\Pr_{S,c}[\phi_S(\psi_c(W)) = \alpha] &= \sum_{j=1}^N \Pr_{S,c}[\psi_c(W) \in \phi_S^{-1}(\alpha) | \psi_c(W) \in T_j] \Pr_{S,c}[\psi_c(W) \in T_j] \\
&= \sum_{j=1}^N \underbrace{\Pr_{S,\mathbf{x}}[\mathbf{x} \in \phi_S^{-1}(\alpha) | \mathbf{x} \in T_j]}_{\text{Lemma 2.3.4}} \Pr_{S,c}[\psi_c(W) \in T_j] \\
&= \sum_{j=1}^{L-1} \Pr_{S,\mathbf{x}}[\phi_S(\mathbf{x}) = \alpha | \mathbf{x} \in T_j] \Pr_{S,c}[\psi_c(W) \in T_j] \\
&\quad + \sum_{j=L}^N \Pr_{S,\mathbf{x}}[\phi_S(\mathbf{x}) = \alpha | \mathbf{x} \in T_j] \Pr_{S,c}[\psi_c(W) \in T_j] \\
&\leq \sum_{j=1}^{L-1} 1 \cdot \Pr_{S,c}[\psi_c(W) \in T_j] \\
&\quad + \sum_{j=L}^N \left(\frac{1}{p} + 2p^{-2} - p^{-3} - p^{-4} \right) \Pr_{S,c}[\psi_c(W) \in T_j] \\
&\leq p^{-2} + \left(\frac{1}{p} + 2p^{-2} - p^{-3} - p^{-4} \right) \sum_{j=L}^N \Pr_{S,c}[\psi_c(W) \in T_j] \\
&\leq p^{-2} + \left(\frac{1}{p} + 2p^{-2} - p^{-3} - p^{-4} \right) \cdot 1 < \frac{1}{p} + 3p^{-2}.
\end{aligned}$$

We complete the proof of Theorem 2.3.3 by noticing two preliminary inequalities similar (and reverse) to the two derived at the beginning of the proof: $\Pr_{S,\mathbf{x}}[\phi_S(\mathbf{x}) = \alpha | \mathbf{x} \in T_j] \geq \left(\frac{1}{p} - p^{-2} \right)$ and $\sum_{j=L}^N \Pr_{S,c}[\psi_c(W) \in T_j] \geq 1 - p^{-2}$, both of which can be derived via a method similar to the one used for the reverse inequalities. Thus, we

have the following lower bound:

$$\begin{aligned}
\Pr_{S,c}[\psi_c(W) \in \phi_S^{-1}(\alpha)] &\geq \sum_{j=L}^N \Pr_{S,c}[\phi_S(\psi_c(W)) = \alpha | \psi_c(W) \in T_j] \Pr_{S,c}[\psi_c(W) \in T_j] \\
&= \sum_{j=L}^N \underbrace{\Pr_{S,\mathbf{x}}[\mathbf{x} \in \phi_S^{-1}(\alpha) | \mathbf{x} \in T_j]}_{\text{Lemma 2.3.4}} \Pr_{S,c}[\psi_c(W) \in T_j] \\
&\geq \left(\frac{1}{p} - p^{-2}\right) \sum_{j=L}^N \Pr_{S,c}[\psi_c(W) \in T_j] \\
&\geq \left(\frac{1}{p} - p^{-2}\right) (1 - p^{-2}) > \frac{1}{p} - 3p^{-2}.
\end{aligned}$$

□

Now we state and prove the ancillary results required by the proof of Theorem 2.3.3.

Lemma 2.3.4 *For any $1 \leq j \leq N$ and any $A \subseteq \Omega_t$,*

$$\Pr_c[\psi_c(W) \in A | \psi_c(W) \in T_j] = \Pr_{\mathbf{x}}[\mathbf{x} \in A | \mathbf{x} \in T_j].$$

Proof. First, define $f: \mathcal{P}(T_j) \rightarrow \mathcal{P}(\mathbf{C})$ so $f(A) = \{c \in \mathbf{C} : \psi_c(W) \in A\}$ (recall that \mathbf{C} is the collection which c is chosen from). Notice that $|f(\{\mathbf{x}\})|$ is constant for all $\mathbf{x} \in T_j$; set $v_j := |f(\{\mathbf{x}\})|$. Also, we note that, for any $U \subseteq T_j$, $|f(U)| = v_j|U|$.

Thus,

$$\begin{aligned}
\Pr_c[\psi_c(W) \in A | \psi_c(W) \in T_j] &= \Pr_c[\psi_c(W) \in A \cap T_j] / \Pr_c[\psi_c(W) \in T_j] \\
&= \frac{|f(A \cap T_j)|}{|\mathbf{C}|} / \frac{|f(T_j)|}{|\mathbf{C}|} \\
&= (v_j|A \cap T_j|) / (v_j|T_j|) \\
&= \frac{|A \cap T_j|}{|\Omega_t|} / \frac{|T_j|}{|\Omega_t|} \\
&= \Pr_{\mathbf{x}}[\mathbf{x} \in A \cap T_j] / \Pr_{\mathbf{x}}[\mathbf{x} \in T_j] \\
&= \Pr_{\mathbf{x}}[\mathbf{x} \in A | \mathbf{x} \in T_j].
\end{aligned}$$

□

Lemma 2.3.5 *If $\varepsilon > 0$, $\delta > 0$, $\alpha \in G$, and $A \subseteq \Omega_t$ with $|A| \geq \frac{1}{p\delta\varepsilon^2}$, then, for all but an δ fraction of $S \in \mathbf{S}$, $\left| \Pr_{\mathbf{x} \in A} [\phi_S(\mathbf{x}) = \alpha] - \frac{1}{p} \right| < \varepsilon$.*

Proof. If, for $S \in \mathbf{S}$, N_S is defined to be $|\phi_S^{-1}(\alpha) \cap A|$, then Corollary 2.2.11 tells us that $\mathbb{E}_{S \in \mathbf{S}}[N_S] = \frac{|A|}{p}$ and $\text{Var}_{S \in \mathbf{S}}[N_S] = \frac{|A|}{p} \left(1 - \frac{1}{p}\right) < \frac{|A|}{p}$. Then, the Chebyshev inequality gives $\Pr_{S \in \mathbf{S}} \left[|N_S - \mathbb{E}_{S \in \mathbf{S}}[N_S]| \geq \sqrt{\frac{|A|}{p\delta}} \right] \leq \frac{\text{Var}_{S \in \mathbf{S}}[N_S]}{\left(\frac{|A|}{p\delta}\right)} < \delta$. Since $\Pr_{\mathbf{x} \in A} [\phi_S(\mathbf{x}) = \alpha] = \frac{N_S}{|A|}$, we see that $\left| \Pr_{\mathbf{x} \in A} [\phi_S(\mathbf{x}) = \alpha] - \frac{1}{p} \right| = \frac{1}{|A|} |N_S - \mathbb{E}_{S \in \mathbf{S}}[N_S]|$. Finally, $\Pr_{S \in \mathbf{S}} \left[\left| \Pr_{\mathbf{x} \in A} [\phi_S(\mathbf{x}) = \alpha] - \frac{1}{p} \right| < \varepsilon \right] = \Pr_{S \in \mathbf{S}} \left[|N_S - \mathbb{E}_{S \in \mathbf{S}}[N_S]| < |A|\varepsilon \right]$, and, by choice of A , we know that this is at least $\Pr_{S \in \mathbf{S}} \left[|N_S - \mathbb{E}_{S \in \mathbf{S}}[N_S]| < \sqrt{\frac{|A|}{p\delta}} \right] > 1 - \delta$. □

Lemma 2.3.6 *If $s = \lceil 13 \log p \rceil$ and $t \geq 24 \log p$, then, for a t -step random walk, we have*

$$\Pr_{c \in \mathbf{C}} \left[|S_s \mathbf{x}| \leq p^5 \mid \mathbf{x} := \psi_c(W) \right] \leq p^{-2}.$$

($S_s \mathbf{x}$ denotes the orbit of \mathbf{x} (under the action of S_s on Ω_t).)

Proof. If $\mathbf{x} = (x_1, x_2, \dots, x_s)$, we may assume (without loss of generality) that $x_1 \leq x_2 \leq \dots \leq x_s$. If r distinct values appear in the \mathbf{x} vector, set $k_1, k_2, \dots, k_r \in \mathbb{N} \setminus \{0\}$ to be the multiplicity of those r values (so $x_1 = x_2 = \dots = x_{k_1} < x_{k_1+1} = \dots = x_{k_1+k_2-1}$, etc.; also, $\sum_i k_i = s$). Notice that $|S_s \mathbf{x}| = \binom{s}{k_1, k_2, \dots, k_r}$.

Fix $u \in \{1, 2, \dots, r\}$ such that

$$\sum_{i=1}^u k_i \geq s/2 > \sum_{i=1}^{u-1} k_i.$$

Set $a := s/2 - \sum_{i=1}^{u-1} k_i$ (so $a \in [1, s/2]$) and set $b := s/2 - \sum_{i=u+1}^r k_i$ (so $b \in [0, s/2]$), making

$$\binom{s}{k_1, k_2, \dots, k_r} = \binom{s}{s/2} \binom{s/2 - a}{k_1, \dots, k_{u-1}} \binom{s/2 - b}{k_{u+1}, \dots, k_r} \binom{s/2}{a} \binom{s/2}{b} \binom{k_u}{a}^{-1};$$

our choice of a , b , and u give us that the latter is at least $\binom{s}{s/2} \cdot 1 \cdot 1 \cdot \binom{s/2}{a} \cdot 1 \cdot \binom{k_u}{a}^{-1}$.

Notice that

$$\binom{s}{s/2} \geq \left(\frac{s}{s/2}\right)^{s/2} = 2^{s/2} \geq p^{13/2} > p^5.$$

Hence, for any $\mathbf{x} \in \Omega_t$, $|S_s \mathbf{x}| > p^5 \binom{s/2}{a} \binom{k_u}{a}^{-1}$. Notice that, if we assume that $k_i \leq s/2$ for all i , we can conclude that $k_u \leq s/2$, so $|S_s \mathbf{x}| > p^5$. Thus,

$$\Pr_{c \in \mathbb{C}}[|S_s \mathbf{x}| \leq p^5 | \mathbf{x} := \psi_c(W)] \leq \Pr_{c \in \mathbb{C}}[\exists i \text{ s.t. } k_i > s/2].$$

We complete the proof by showing that the latter probability is no more than p^{-2} .

Using the inequality (from [38]) $\binom{i}{\lambda i} \leq \frac{2^{H(\lambda)i}}{\sqrt{2\pi\lambda(1-\lambda)i}}$, we see that

$$\binom{s}{s/2} \leq \frac{2^{H(1/2)s}}{\sqrt{2\pi\frac{1}{2}(1-\frac{1}{2})s}} = \frac{2^s}{(\pi s/2)^{1/2}} \leq \frac{1}{5}p^8,$$

so

$$\begin{aligned} \Pr_{c \in \mathbb{C}}[\exists i \text{ s.t. } k_i > s/2] &= \sum_{\theta=0}^{\lfloor \frac{t}{s/2+1} \rfloor} \Pr_{c \in \mathbb{C}}[\text{at least } s/2 + 1 \text{ of the } k_i \text{s equal } \theta] \\ &\leq \sum_{\theta=0}^{\lfloor \frac{t}{s/2} \rfloor} \Pr_{c \in \mathbb{C}}[\text{at least } s/2 \text{ of the } k_i \text{s equal } \theta] \\ &\leq \sum_{\theta=0}^{\lfloor \frac{t}{s/2} \rfloor} \binom{s}{s/2} \Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = \theta] \\ &\leq \frac{1}{5}p^8 \sum_{\theta=0}^{\lfloor \frac{t}{s/2} \rfloor} \Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = \theta]. \end{aligned}$$

The final section of this proof shows that $\Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = \theta] \leq 5p^{-10} \frac{s}{4t}$ (which is less than $5p^{-10} \frac{1}{\lfloor 2t/s \rfloor + 1}$) for four different cases: $\theta = 0$, $\theta = 1$, $\theta = 2t/s$ (only relevant when $s|(2t)$), and all other θ used in the sum ($\theta \in \{2, 3, \dots, \lfloor 2t/s \rfloor - 1\}$).

In any case,

$$\begin{aligned} \Pr_{c \in \mathbb{C}}[x_1 = \dots = x_{s/2} = \theta] &= \binom{t}{\theta, \dots, \theta, t - (s/2)\theta} \left(\left(\frac{1}{s} \right)^\theta \right)^{s/2} \left(1 - \frac{s/2}{s} \right)^{t - (s/2)\theta} \\ &= \left(\frac{t!}{(\theta!)^{s/2} (t - (s/2)\theta)!} \right) \left(\frac{2}{s} \right)^{(s/2)\theta} \left(\frac{1}{2} \right)^t. \end{aligned}$$

In the first three cases, we first show that $\Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = \theta] \leq (2t/s)^{s/2} 2^{-t}$.

In case $\theta = 0$, we calculate

$$\Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = 0] = 2^{-t} \leq \left(\frac{2t}{s} \right)^{s/2} 2^{-t}.$$

In case $\theta = 1$,

$$\Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = 1] = \frac{t!}{(t - (s/2))!} \left(\frac{1}{s/2} \right)^{s/2} \left(\frac{1}{2} \right)^t < \left(\frac{2t}{s} \right)^{s/2} 2^{-t}.$$

In case $\theta = 2t/s$, we recall that for a positive integer α , $\sqrt{2\pi\alpha} \left(\frac{\alpha}{e} \right)^\alpha < \alpha! < \sqrt{2\pi\alpha} \left(\frac{\alpha}{e} \right)^\alpha e^{\frac{1}{12\alpha}}$. Hence,

$$\begin{aligned} \Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = 2t/s] &= \frac{t!}{((2t/s)!)^{s/2}} \left(\frac{1}{s} \right)^t \\ &< \frac{\sqrt{2\pi t} \left(\frac{t}{e} \right)^t e^{\frac{1}{12t}}}{\left(\sqrt{2\pi(2t/s)} \left(\frac{2t/s}{e} \right)^{2t/s} \right)^{s/2}} \left(\frac{1}{s} \right)^t \\ &= (2\pi t)^{\frac{1}{2} - \frac{s}{4}} e^{\frac{1}{12t}} (s/2)^{s/4} 2^{-t} \leq \left(\frac{2t}{s} \right)^{s/2} 2^{-t}. \end{aligned}$$

Now notice that

$$\left(\frac{2t}{s} \right)^{s/2} 2^{-t} \leq \left(\frac{t}{t/e} \right)^{t/e} 2^{-t} = 2^{-(\log 2e^{-1/e})t} \leq 2^{-11 \log p},$$

since $t \geq 24 \log p$.

Hence, we deduce that in the first three cases ($\theta = 0$, $\theta = 1$, and $\theta = 2t/s$),

$$\Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = \theta] \leq p^{-11} \leq \frac{5s}{4} p^{-10} \frac{1}{p} < 5p^{-10} \frac{s}{4t},$$

as claimed.

We now show the inequality for the remaining θ (integer θ such that $2 \leq \theta < 2t/s$), utilizing the inequalities for $\alpha!$ that we used in the $\theta = 2t/s$ case:

$$\begin{aligned} \Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = \theta] &= \left(\frac{t!}{(\theta!)^{s/2} (t - (s/2)\theta)!} \right) \left(\frac{1}{s/2} \right)^{(s/2)\theta} \left(\frac{1}{2} \right)^t \\ &< \frac{\sqrt{2\pi t} \left(\frac{t}{e} \right)^t e^{\frac{1}{12t}}}{\left(\sqrt{2\pi\theta} \left(\frac{\theta}{e} \right)^\theta \right)^{s/2} \sqrt{2\pi(t - s\theta/2)} \left(\frac{t - s\theta/2}{e} \right)^{t - s\theta/2}} \\ &\quad \cdot \left(\frac{1}{s/2} \right)^{s\theta/2} \left(\frac{1}{2} \right)^t \\ &= \sqrt{\frac{t}{t - s\theta/2}} e^{\frac{1}{12t}} (2\pi\theta)^{-s/4} \left(\frac{t}{2(t - s\theta/2)} \right)^t \\ &\quad \cdot \left(\frac{t - s\theta/2}{s\theta/2} \right)^{s\theta/2} \\ &\leq \sqrt{\frac{t}{t - s\theta/2}} e^{\frac{1}{12t}} (2\pi\theta)^{-s/4} \\ &\leq \frac{5}{2} \sqrt{\frac{t}{t - s\theta/2}} (2\pi\theta)^{-s/4} \\ &\leq \frac{5}{2} \sqrt{\frac{t}{t - s\theta/2}} (2\pi(2))^{-s/4} \leq \frac{5}{2} \sqrt{t} (4\pi)^{-s/4}, \end{aligned}$$

since s does not divide $2t$ (so $t - s\theta/2 \geq 1$).

Further, as $s \leq 13 \log p$,

$$\begin{aligned} \Pr_{c \in \mathbb{C}}[x_1 = x_2 = \dots = x_{s/2} = \theta] &\leq \frac{5}{2} \sqrt{t} (4\pi)^{-s/4} \\ &= \left(5p^{-10} \frac{s}{4t}\right) p^{10} t^{3/2} (s/2)^{-1} p^{-\frac{s}{\log p} \log(4\pi)/4} \\ &\leq \left(5p^{-10} \frac{s}{4t}\right) p^{10} p^{3/2} p^{-13 \log(4\pi)/4} < 5p^{-10} \frac{s}{4t}. \end{aligned}$$

□

2.3.4 Pseudorandomness

While the rest of this chapter focuses on using a c with limited independence to allow us to need only constant space, we could instead use a totally independent c chosen pseudorandomly. While this alleviates the need to the VGCD assumption, it adds a reliance upon a complexity assumption. (The VGCD assumption seems more likely to be provable than complexity assumptions upon which pseudorandom functions are based (*e.g.*, that factoring is hard).)

In addition to relying on a complexity-theoretic assumption, not having to work with limited-independence functions has another key drawback. The function must be evaluated at $\tilde{O}(\sqrt{p})$ points, a parameter that will affect the run time of the generator (*e.g.*, by needing to work in a large group).

2.4 Proof of Theorem 2.2.4 (Degeneracy)

Here we restate Theorem 2.2.4, then proceed to prove it:

Theorem 2.2.4 (Degeneracy) *If the Cayley rho cycle-finding subroutine FIND-CYCLE returns successfully, CR-DLOG will solve DLOG with probability at least $2/s^2$; thus, the expected number of calls to the cycle-finding algorithm FIND-CYCLE is no more than $s^2/2$.*

Proof. Recall that CR-DLOG constructs $S \subseteq G$ by first randomly picking $\mathbf{r} = (r_1, \dots, r_s) \in \mathbb{Z}_p^s$, and then setting $S = (g_1, \dots, g_s)$ to be equal to a random permutation of $(g^{r_1}, \dots, g^{r_{s/2}}, y^{r_{s/2+1}}, \dots, y^{r_s})$. For notational convenience, we define the

function $h: \mathbb{Z}_p^s \times S_s \rightarrow G^s$ (where S_s is the group of all permutations on s elements) as $(\mathbf{r}, \sigma) \mapsto (\gamma_{\sigma(1)}, \dots, \gamma_{\sigma(s)})$, where $\gamma_i = \begin{cases} g^{r_i} & 1 \leq i \leq s/2 \\ y^{r_i} & \text{else} \end{cases}$. Now notice that, utilizing this new notation, we may say that CR-DLOG chooses S by randomly choosing $\mathbf{r} \in \mathbb{Z}_p^s$ and randomly choosing $\sigma \in S_s$ and setting $S := h(\mathbf{r}, \sigma)$.

Let FIND-CYCLE(\mathcal{G}) find a length- t cycle with a path type \mathbf{w} . From this, we get an equation of the form $z_0 = z_0 \prod_{i=1}^s g_i^{w_i}$, for some initial node $z_0 \in G$ and $0 \leq w_i \leq t$, where $\sum_{i=1}^s w_i = t$ (in other words, \mathbf{w} is the path type of the cycle). From the definition of the g_i , we see that $\prod_{i=1}^{s/2} g^{-r_i w_i} = \prod_{i=s/2+1}^s y^{r_i w_i}$. Hence, $-\sum_{i=1}^{s/2} r_i w_i = x \sum_{i=s/2+1}^s r_i w_i \pmod{p}$, which yields x unless $\sum_{i=1}^{s/2} r_i w_i = 0 \pmod{p}$. The probability that we *cannot* find x is

$$\Pr_{\mathbf{r}, \sigma} \left[\sum_{i=1}^{s/2} r_i w_i \equiv 0 \pmod{p} \right] = \sum_{\bar{S} \in \mathcal{S}} \Pr_{\mathbf{r}, \sigma} \left[\sum_{i=1}^{s/2} r_i w_i \equiv 0 \pmod{p} \mid h(\mathbf{r}, \sigma) = \bar{S} \right] \Pr_{\mathbf{r}, \sigma} [h(\mathbf{r}, \sigma) = \bar{S}] .$$

To help calculate this probability, we define $f_i: \mathcal{S} \times S_s \rightarrow \mathbb{Z}_p$ for $1 \leq i \leq s/2$ as $(S, \sigma) \mapsto \log_g g_{\sigma^{-1}(i)}$ and for $s/2 + 1 \leq i \leq s$ as $(S, \sigma) \mapsto x^{-1} \log_g g_{\sigma^{-1}(i)}$. Notice that f_i is constructed exactly so $f_i(h(\mathbf{r}, \sigma), \sigma) = r_i$. Thus,

$$\begin{aligned} \Pr_{(\mathbf{r}, \sigma) \in h^{-1}(\bar{S})} \left[\sum_{i=1}^{s/2} r_i w_i \equiv 0 \pmod{p} \right] &= \Pr_{\sigma \in S_s} \left[\sum_{i=1}^{s/2} f_i(\bar{S}, \sigma) w_i \equiv 0 \pmod{p} \right] \\ &= \Pr_{\sigma \in S_s} \left[\sum_{i=1}^{s/2} w_i \log_g g_{\sigma^{-1}(i)} \equiv 0 \pmod{p} \right] , \end{aligned}$$

which, by taking $k_j := \log_g g_j$, Lemma 2.4.1 tells us is at most $1 - 2/s^2$.

Hence, we have an upper bound on the probability that we must rerun FIND-CYCLE:

$$\Pr_{\mathbf{r}, \sigma} \left[\sum_{i=1}^{s/2} r_i w_i \equiv 0 \pmod{p} \right] \leq \sum_{\bar{S} \in \mathcal{S}} \left(1 - \frac{2}{s^2} \right) \Pr_{\mathbf{r}, \sigma} [h(\mathbf{r}, \sigma) = \bar{S}] = 1 - \frac{2}{s^2} ;$$

thus, we expect to rerun FIND-CYCLE at most $s^2/2$ times. \square

Lemma 2.4.1 *If $k_1, k_2, \dots, k_s \in \mathbb{Z}_p^\times$ are such that, for $i \neq j$, $k_i \neq \pm k_j \pmod{p}$; $t \in \mathbb{Z}_p$; and $\mathbf{w} \in \mathbb{Z}_p^s \setminus \{\mathbf{0}\}$, then*

$$\Pr_{\sigma \in S_s} \left[\sum_{i=1}^{s/2} k_{\sigma^{-1}(i)} w_i = \sum_{j=s/2+1}^s k_{\sigma^{-1}(j)} w_j = 0 \pmod{p} \right] \leq 1 - \frac{2}{s^2}.$$

Proof. Set $A := \left\{ \sigma \in S_s : \sum_{i=1}^{s/2} k_{\sigma^{-1}(i)} w_i = \sum_{j=s/2+1}^s k_{\sigma^{-1}(j)} w_j = 0 \pmod{p} \right\}$. Also, for notational convenience, set $z_\sigma(i) := k_{\sigma^{-1}(i)} w_i$.

We now show that, for every $\sigma \in A$, we can choose a_σ, b_σ such that $1 \leq a_\sigma \leq s/2 < b_\sigma \leq s$ and $z_\sigma(a_\sigma) \neq z_\sigma(b_\sigma)$. To do so, we assume that we *cannot* choose such a_σ, b_σ . Notice that, then, $z_\sigma(i) = c_\sigma$ for all i (for some constant c_σ), so, since $\sigma \in A$, $0 = \sum_{i=1}^{s/2} z_\sigma(i) = c_\sigma \cdot (s/2) \pmod{p}$. Thus, $p|c_\sigma(s/2)$, so $p|c_\sigma$ or $p|(s/2)$. As $0 < s/2 < p$, we conclude that $p|c_\sigma$ (so $c_\sigma = 0 \pmod{p}$) and, hence, $k_{\sigma^{-1}(i)} w_i = z_\sigma(i) = c_\sigma = 0 \pmod{p}$ for all i . But, as the $k_{\sigma^{-1}(i)}$ are nonzero, this means that $w_{\sigma(i)} = 0$ for all i , a contradiction.

Hence, for any $\sigma \in A$, we may define a canonical a_σ, b_σ as described above. For a fixed $\sigma \in A$, set $\sigma' := \sigma \circ (a_\sigma b_\sigma)$. Notice that, since $\sigma \in A$, we know that

$$\sum_{i=1}^{s/2} z_{\sigma'}(i) = \left(\sum_{i=1}^{s/2} z_\sigma(i) \right) - z_\sigma(a_\sigma) + z_\sigma(b_\sigma) = z_\sigma(b_\sigma) - z_\sigma(a_\sigma),$$

which is nonzero by choice of a_σ and b_σ . Thus, $\sigma' \notin A$. Define $f: A \rightarrow S_s \setminus A$ so $f(\sigma) = \sigma \circ (a_\sigma b_\sigma)$. Notice that for any $\sigma' \in S_s \setminus A$, $|f^{-1}(\sigma')| \leq (s/2)^2 \leq s^2/2 - 1$; hence, $|A| \leq (s^2/2 - 1) |S_s \setminus A|$. Now we conclude that

$$\begin{aligned} \Pr_{\sigma \in S_s} \left[\sum_{i=1}^{s/2} k_{\sigma^{-1}(i)} w_i = \sum_{j=s/2+1}^s k_{\sigma^{-1}(j)} w_j = 0 \pmod{p} \right] &= \frac{|A|}{|S_s|} = \frac{(s^2/2) |A|}{(s^2/2) |S_s|} \\ &\leq \frac{(s^2/2 - 1) |A| + (s^2/2 - 1) |S_s \setminus A|}{(s^2/2) |S_s|} = 1 - \frac{2}{s^2}. \end{aligned}$$

□

2.5 Proof of Theorem 2.2.5 (Cycle-Finding Time)

Here we restate Theorem 2.2.5, then proceed to prove it:

Theorem 2.2.5 (Cycle-Finding Time) *Let CR-DLOG take $(8 \log p)\sqrt{p}$ moves on the graph. Then, under the VGCD assumption, the probability (over the random choices made by CR-DLOG) of a nontrivial cycle occurring is at least $\frac{1}{2}$.*

Proof. Let $z_0, z_1, \dots, z_t \in G$ denote the sequence produced by CR-DLOG. Define the random variables

$$\chi_{ij} = \begin{cases} 0 & \text{if } z_i \neq z_j \\ 1 & \text{if } z_i = z_j \end{cases}$$

for $i, j \in \{0, 1, \dots, t\}$. Then the number of collisions in the first t steps is $\chi := \sum_{0 \leq i < j \leq t} \chi_{ij}$. Put $\mu = E_{S,c}[\chi]$ and $\sigma^2 = E_{S,c}[(\chi - \mu)^2]$. We wish to bound $\Pr_{S,c}[\chi = 0] \leq \Pr_{S,c}[|\chi - \mu| \geq \mu] \leq \frac{\sigma^2}{\mu^2} = \frac{E_{S,c}[\chi^2]}{\mu^2} - 1$.

First, we note that Corollary 2.5.1 tells us that

$$\mu = E_{S,c}[\chi] = \sum_{i < j} E_{S,c}[\chi_{ij}] = \sum_{i < j} \Pr_{S,c}[\chi_{ij} = 1] = \binom{t+1}{2} \frac{1}{p}.$$

Now notice that

$$E_{S,c}[\chi^2] = \sum_{i < j} \sum_{k < \ell} \Pr_{S,c}[\chi_{ij} \chi_{k\ell} = 1],$$

which we now prove is at most $\binom{t+1}{2}^2 \cdot \frac{3}{2p^2}$.

For notational convenience, define

$$\begin{aligned} \mathcal{U} &:= \{(i, j, k, \ell) \in \{0, 1, \dots, t\}^4 : i < j, k < \ell\}, \\ \mathcal{U}_1 &:= \{(i, j, k, \ell) \in \mathcal{U} : (\ell - k \leq \log p) \wedge (j - i \leq \log p)\}, \\ \mathcal{U}_2 &:= \{(i, j, k, \ell) \in \mathcal{U} : \ell - k \neq j - i\}, \text{ and} \\ \mathcal{U}_3 &:= \left\{ (i, j, k, \ell) \in \mathcal{U} : |k - i| > \frac{1}{2} \log p \right\}. \end{aligned}$$

Notice that we can partition \mathcal{U} into $\mathcal{U}_1, \overline{\mathcal{U}}_1 \cap \mathcal{U}_2, \overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \mathcal{U}_3,$ and $\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \overline{\mathcal{U}}_3$.

Thus,

$$\begin{aligned} E_{S,c}[\chi^2] &= \sum_{\mathcal{U}_1} \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1] + \sum_{\overline{\mathcal{U}_1} \cap \mathcal{U}_2} \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1] \\ &\quad + \sum_{\overline{\mathcal{U}_1} \cap \mathcal{U}_2 \cap \mathcal{U}_3} \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1] + \sum_{\overline{\mathcal{U}_1} \cap \mathcal{U}_2 \cap \overline{\mathcal{U}_3}} \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1]. \end{aligned}$$

Now we bound each term. First, since Lemmata 2.5.2 and 2.5.3 tell us that for all $(i, j, k, \ell) \in \mathcal{U}$, $\Pr_{S,c}[\chi_{ij}\chi_{kl} = 1] \leq 1/p$, we know that

$$\sum_{\mathcal{U}_1} \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1] \leq \sum_{\mathcal{U}_1} \frac{1}{p} = |\mathcal{U}_1| \frac{1}{p} \leq \frac{1}{4} (2t+1 - \log p)^2 (\log^2 p) \frac{1}{p} \leq \binom{t+1}{2} \frac{1}{12p^2}.$$

Let $\mathcal{E} = \mathcal{E}(\mathbf{x}, \mathbf{y})$ denote the event that $\text{vgcd}(\mathbf{x}) = 1$ and $\text{vgcd}(\mathbf{y}) = 1$. We will now see that, given \mathcal{E} and $(i, j, k, \ell) \in \mathcal{U}_2$, \mathbf{x} and \mathbf{y} are linearly independent. Notice that when \mathbf{x} and \mathbf{y} are linearly dependent, we may choose relatively prime $\alpha, \beta \in \mathbb{Z}$ such that $\alpha\mathbf{x} = \beta\mathbf{y}$. Then $\alpha|y_a$ for all a , so $\alpha|\text{vgcd}(\mathbf{y})$. Similarly, $\beta|\text{vgcd}(\mathbf{x})$. Given \mathcal{E} , we would conclude that $\alpha, \beta \in \{-1, 1\}$. Since all the entries of \mathbf{x} and \mathbf{y} are nonnegative, α and β must then have the same sign. Hence, $\mathbf{x} = \mathbf{y}$. But $(i, j, k, \ell) \in \mathcal{U}_2$, so $\|\mathbf{y}\| = \ell - k \neq j - i = \|\mathbf{x}\|$, contradicting that $\mathbf{x} = \mathbf{y}$.

Now we now know that, given \mathcal{E} and $(i, j, k, \ell) \in \mathcal{U}_2$, \mathbf{x} and \mathbf{y} are linearly independent; therefore we know (by Lemma 2.5.2) that $\Pr_{S,c}[\chi_{ij}\chi_{kl} = 1 \mid \mathcal{E}] = 1/p^2$ for $(i, j, k, \ell) \in \mathcal{U}_2$. Thus,

$$\begin{aligned} \sum_{\overline{\mathcal{U}_1} \cap \mathcal{U}_2} \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1] &= \sum_{\overline{\mathcal{U}_1} \cap \mathcal{U}_2} \left(\Pr_{S,c}[\chi_{ij}\chi_{kl} = 1 \mid \mathcal{E}] \Pr_{S,c}[\mathcal{E}] + \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1 \mid \overline{\mathcal{E}}] \Pr_{S,c}[\overline{\mathcal{E}}] \right) \\ &\leq \sum_{\overline{\mathcal{U}_1} \cap \mathcal{U}_2} \left(\frac{1}{p^2} \cdot 1 + \frac{1}{p} \cdot 2 \left(\frac{1}{8p} \right) \right), \end{aligned}$$

from two applications of the VGCD assumption. Thus,

$$\sum_{\overline{\mathcal{U}_1} \cap \mathcal{U}_2} \Pr_{S,c}[\chi_{ij}\chi_{kl} = 1] = |\overline{\mathcal{U}_1} \cap \mathcal{U}_2| \frac{5}{4p^2} \leq |\mathcal{U}| \frac{5}{4p^2} = \binom{t+1}{2} \frac{5}{4p^2}.$$

When $(i, j, k, \ell) \in \overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \mathcal{U}_3$, set $\mathbf{z} := \psi_c(\{i+1, \dots, k\} \cup \{j+1, \dots, \ell\})$. Let $\mathcal{E}' = \mathcal{E}'(\mathbf{x}, \mathbf{y}, \mathbf{z})$ denote the event that $\text{vgcd}(\mathbf{x}) = 1$, $\text{vgcd}(\mathbf{y}) = 1$, and $\text{vgcd}(\mathbf{z}) = 1$.

We will now see that, given \mathcal{E}' , \mathbf{x} and \mathbf{y} are linearly independent. Notice that when \mathbf{x} and \mathbf{y} are linearly dependent, we may choose relatively prime $\alpha, \beta \in \mathbb{Z}$ such that $\alpha\mathbf{x} = \beta\mathbf{y}$. So $\alpha | \text{vgcd}(\mathbf{y})$ and $\beta | \text{vgcd}(\mathbf{x})$. Given \mathcal{E}' , we would conclude that $\alpha, \beta \in \{-1, 1\}$. Since all the entries of \mathbf{x} and \mathbf{y} are nonnegative, α and β must then have the same sign. Hence, $\mathbf{x} = \mathbf{y}$. Without loss of generality, $i < k < j < \ell$. Define $\mathbf{z}^{(A)} := \psi_c(\{i+1, \dots, k\})$, $\mathbf{z}^{(B)} := \psi_c(\{k+1, \dots, j\})$, $\mathbf{z}^{(C)} := \psi_c(\{j+1, \dots, \ell\})$. Now notice that $\mathbf{x} = \mathbf{z}^{(A)} + \mathbf{z}^{(B)}$ and $\mathbf{y} = \mathbf{z}^{(B)} + \mathbf{z}^{(C)}$. Thus, $\mathbf{x} = \mathbf{y}$ tells us that $\mathbf{z}^{(A)} = \mathbf{z}^{(C)}$. But $\mathbf{z} = \mathbf{z}^{(A)} + \mathbf{z}^{(C)}$, so $2 | \text{vgcd}(\mathbf{z})$, contradicting \mathcal{E}' .

Now we now know that, given \mathcal{E}' , \mathbf{x} and \mathbf{y} are linearly independent; therefore we know (by Lemma 2.5.2) that $\Pr_{S,c}[\chi_{ij}\chi_{k\ell} = 1 \mid \mathcal{E}'] = 1/p^2$. Thus,

$$\begin{aligned} \sum_{\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \mathcal{U}_3} \Pr_{S,c}[\chi_{ij}\chi_{k\ell} = 1] &= \sum_{\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \mathcal{U}_3} \left(\Pr_{S,c}[\chi_{ij}\chi_{k\ell} = 1 \mid \mathcal{E}'] \Pr_{S,c}[\mathcal{E}'] \right. \\ &\quad \left. + \Pr_{S,c}[\chi_{ij}\chi_{k\ell} = 1 \mid \overline{\mathcal{E}'}] \Pr_{S,c}[\overline{\mathcal{E}'}] \right) \\ &\leq \sum_{\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \mathcal{U}_3} \left(\frac{1}{p^2} \cdot 1 + \frac{1}{p} \cdot 3 \left(\frac{1}{8p} \right) \right), \end{aligned}$$

from three applications of the VGCD assumption. Thus,

$$\begin{aligned} \sum_{\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \mathcal{U}_3} \Pr_{S,c}[\chi_{ij}\chi_{k\ell} = 1] &= |\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \mathcal{U}_3| \frac{11}{8p^2} \leq |\overline{\mathcal{U}}_2| \frac{11}{8p^2} = \frac{t(t+1)(2t+1)}{6} \cdot \frac{11}{8p^2} \\ &\leq \binom{t+1}{2} \frac{1}{12p^2}. \end{aligned}$$

Lastly,

$$\begin{aligned} \sum_{\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \overline{\mathcal{U}}_3} \Pr_{S,c}[\chi_{ij}\chi_{k\ell} = 1] &\leq \sum_{\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \overline{\mathcal{U}}_3} \frac{1}{p} = |\overline{\mathcal{U}}_1 \cap \overline{\mathcal{U}}_2 \cap \overline{\mathcal{U}}_3| \frac{1}{p} \leq |\overline{\mathcal{U}}_2 \cap \overline{\mathcal{U}}_3| \frac{1}{p} \\ &< \binom{t+1}{2} \left(2 \cdot \frac{\log p}{2} + 1 \right) \leq \binom{t+1}{2} \frac{1}{12p^2}. \end{aligned}$$

Tying all four bounds back together, we see that

$$E_{S,c}[\chi^2] \leq \binom{t+1}{2}^2 \left(\frac{1}{12p^2} + \frac{5}{4p^2} + \frac{1}{12p^2} + \frac{1}{12p^2} \right) = \binom{t+1}{2}^2 \frac{3}{2p^2}.$$

Hence,

$$\Pr_{S,c}[\chi = 0] \leq \frac{E_{S,c}[\chi^2]}{\mu^2} - 1 \leq \frac{\binom{t+1}{2}^2 \cdot \frac{3}{2p^2}}{\left(\binom{t+1}{2} \frac{1}{p}\right)^2} - 1 = \frac{1}{2}.$$

□

Here we note a corollary of Lemma 2.2.8, after which we present various lemmata needed throughout this section.

Corollary 2.5.1 *For all i, j such that $0 \leq i < j \leq t$ and for all c ,*

$$\Pr_{S \in \mathcal{S}}[\chi_{ij} = 1] = \frac{1}{p}.$$

Proof. Set $\mathbf{x} := \psi_c(\{i+1, i+2, \dots, j\}) \in \mathbb{Z}_p^s$. Then, we have that $\Pr_{S \in \mathcal{S}}[\chi_{ij} = 1] = \Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = 1]$, which Lemma 2.2.8 tells us is $\frac{1}{p}$. □

Lemma 2.5.2 *For all c, i, j, k, ℓ such that $0 \leq i < j \leq t$ and $0 \leq k < \ell \leq t$, if $\mathbf{x} := \psi_c(\{i+1, i+2, \dots, j\}) \in \mathbb{Z}_p^s$ and $\mathbf{y} := \psi_c(\{k+1, k+2, \dots, \ell\}) \in \mathbb{Z}_p^s$ are linearly independent, then*

$$\Pr_{S \in \mathcal{S}}[\chi_{ij}\chi_{k\ell} = 1] = \frac{1}{p^2}.$$

Proof. Since $z_i = z_j$ (and, thus, $\chi_{ij} = 1$) exactly when $\phi_S(\mathbf{x}) = 1$ (similarly, $\chi_{k\ell} = 1 \Leftrightarrow \phi_S(\mathbf{y}) = 1$), we can conclude that

$$\Pr_{S \in \mathcal{S}}[\chi_{ij}\chi_{k\ell} = 1] = \Pr_{S \in \mathcal{S}}[\phi_S(\mathbf{x}) = 1 \wedge \phi_S(\mathbf{y}) = 1],$$

which, by Lemma 2.2.9, is seen to be $\frac{1}{p^2}$. □

Lemma 2.5.3 *For all c, i, j, k, ℓ such that $0 \leq i < j \leq t$ and $0 \leq k < \ell \leq t$, if $\mathbf{x} := \psi_c(\{i+1, i+2, \dots, j\}) \in \mathbb{Z}_p^s$ and $\mathbf{y} := \psi_c(\{k+1, k+2, \dots, \ell\}) \in \mathbb{Z}_p^s$ are linearly*

dependent, then

$$\Pr_{S \in \mathcal{S}}[\chi_{ij}\chi_{kl} = 1] = \frac{1}{p} .$$

Proof. Clearly $\chi_{ij}\chi_{kl} = 1 \Rightarrow \chi_{ij} = 1$; we show that, when \mathbf{x} and \mathbf{y} are linearly dependent, $\chi_{ij} = 1 \Rightarrow \chi_{ij}\chi_{kl} = 1$.

When $\chi_{ij} = 1$, $\phi_S(\mathbf{x}) = 1$, so

$$\phi_S(\mathbf{y})^{t_{\mathbf{x}}} = \phi_S(t_{\mathbf{x}}\mathbf{y}) = \phi_S(t_{\mathbf{y}}\mathbf{x}) = \phi_S(\mathbf{x})^{t_{\mathbf{y}}} = 1 ,$$

therefore, since $0 < t_{\mathbf{x}} < p$, and $\phi_S(\mathbf{y}) \in G$ (which has prime order p), we conclude that $\phi_S(\mathbf{y}) = 1$ and, hence $\chi_{kl} = 1$. Thus, $\chi_{ij} = 1 \Rightarrow \chi_{ij}\chi_{kl} = 1$, so $\chi_{ij} = 1 \Leftrightarrow \chi_{ij}\chi_{kl} = 1$, allowing us to conclude

$$\Pr_{S \in \mathcal{S}}[\chi_{ij}\chi_{kl} = 1] = \Pr_{S \in \mathcal{S}}[\chi_{ij} = 1] = \frac{1}{p} ,$$

by Corollary 2.5.1. □

2.6 Secure Hash Functions

Earlier, Bellare and Micciancio [8] presented a DLOG-based construction and analyzed it in the random-oracle model. However, in real-world implementations, a break in their hash functions need not give a break in DLOG. We now present a DLOG-based construction whose break would lead to a break in DLOG.

For work related to hash functions using girth (but without any analysis of security), see [1, 18, 58, 59]. Our approach begins with a presumably hard cycle-finding problem (*i.e.*, to algorithms with limited resources, the girth appears exponentially larger than its actual value) from which one constructs secure hash functions; in a way, our approach strengthens these earlier works' security. Also, similar to hash functions in other papers (*e.g.*, [7, 24, 26]), our hash function is incremental; *i.e.*, if a portion of an input is changed, the hash function can be recomputed on the new input from the old hash value, in time proportional to the change and not the entire input.

Fix a group G of order p and a generator $g \in G$ with respect to which we will find discrete logarithms. Fix $s = \Omega(\log p)$. For $S := \{g_1, g_2, \dots, g_s\} \subseteq G$, we define $F_S: \{1, 2, \dots, s\}^* \rightarrow G$ as $(a_1, a_2, \dots, a_t) \mapsto \prod_{i=1}^t g_{a_i}^{2^{t-i}}$. We assume that the inputs are buffered in the following sense: all our inputs to F_S will have a fixed length $L < p$, where we define the length of (a_1, a_2, \dots, a_t) as the corresponding length of the path on the Cayley graph (relative to fixed S) by viewing the computation of F_S as a walk on the graph in the natural way. If $F_S(a_1, a_2, \dots, a_t) = \prod_{r=1}^n g_r^{w_r}$, then its length is $\|\mathbf{w}\| = \sum_r w_r$ (which happens to equal $2^t - 1$).

Now we show that, if DLOG is hard, then the hash function family $\{F_S\}_S$ defined above is collision resistant.

Theorem 2.6.1 *If \mathcal{A} is an algorithm that, when given S , produces two inputs that correspond to nontrivial graph paths of fixed length $L < p$ (i.e., $t \leq \log p$), then there exists an algorithm \mathcal{B} that finds x given $y = g^x$ that runs in time $O(s^2 \cdot \text{TIME}(\mathcal{A}))$.*

Proof. We construct an algorithm $\mathcal{B}(y)$ for computing x as follows: $\mathcal{B}(y)$ first randomly chooses r_1, r_2, \dots, r_s from \mathbb{Z}_p . Define $S := \{g_1, g_2, \dots, g_s\}$ be a random permutation of $\{g^{r_1}, g^{r_2}, \dots, g^{r_{s/2}}, y^{r_{s/2+1}}, y^{r_{s/2+2}}, \dots, y^{r_s}\}$. Give the input S to \mathcal{A} and denote the output (two colliding inputs) by $\mathbf{a} = (a_1, a_2, \dots, a_t)$ and $\mathbf{b} = (b_1, b_2, \dots, b_m)$, where $a_i, b_i \in \{1, 2, \dots, s\}$.

Notice that $F_S(\mathbf{a}) = \prod_{i=1}^t g_{a_i}^{2^{t-i}} = \prod_{i=1}^s g_i^{\gamma_i}$, where we define $\gamma_i := \sum_{j \in V_i} 2^{t-j}$, $V_i := \{j \leq t : a_j = i\}$. Similarly, $F_S(\mathbf{b}) = \prod_{i=1}^m g_{b_i}^{2^{m-i}} = \prod_{i=1}^s g_i^{\eta_i}$, where $\eta_i := \sum_{j \in W_i} 2^{m-j}$, $W_i := \{j \leq m : b_j = i\}$.

Since $F_S(\mathbf{a}) = F_S(\mathbf{b})$, we get $\prod_{i=1}^{s/2} g^{r_i(\gamma_i - \eta_i)} = \prod_{i=s/2+1}^s y^{-r_i(\gamma_i - \eta_i)}$; hence, we know that $\sum_{i=1}^{s/2} r_i(\gamma_i - \eta_i) = -x \sum_{i=s/2+1}^s r_i(\gamma_i - \eta_i) \pmod{p}$.

We know the r_i and s_i and can compute the γ_i and η_i , so, unless $\sum_{i=s/2+1}^s r_i(\gamma_i - \eta_i) = 0 \pmod{p}$, we can find x ; otherwise we simply repeat the procedure with new randomly constructed S .

Define \mathbf{w} so $w_i = \gamma_i - \eta_i$ and notice that, since $\mathbf{a} \neq \mathbf{b}$, $w_i \neq 0$ for at least one i . Thus, Lemma 2.4.1 tells us that we expect to repeat this procedure at most $\frac{s^2}{2}$ times. \square

2.7 Experimental Results

Experiments were performed (with much help from the NTL [53] number-theory package) to compute, for various b and various s , the average (over M_1 different b -bit primes and M_2 different sets of generators of size s) value for the ratio t/\sqrt{p} (where t is the number of steps required to find a discrete logarithm and $p = |G|$):

b	s	M_1	M_2	t/\sqrt{p}
9	10	100	100	1.92815
	18	100	100	1.74489
	36	100	100	1.64414
	72	100	100	1.57033
13	14	100	100	1.75660
	26	100	100	1.58203
	52	100	100	1.54226
	104	100	100	1.50104
16	16	100	100	1.80298
	32	100	100	1.60731
	64	100	100	1.50684
	128	100	100	1.48697
19	20	100	100	1.77065
	38	100	100	1.53030
	76	100	100	1.49859
	156	100	100	1.48713
23	24	100	100	1.84696
	46	100	100	1.51326
	92	100	100	1.49586
	184	100	100	1.49537
26	26	80	40	1.57050
	52	80	40	1.51372
	104	80	40	1.48415
	208	80	40	1.48644

b	s	M_1	M_2	t/\sqrt{p}
29	30	40	30	1.97471
	58	40	30	1.51190
	116	40	30	1.49100
	232	40	30	1.49889
33	34	30	30	1.51474
	66	30	30	1.49360
	132	30	30	1.49982
	264	30	30	1.51298
36	36	25	20	1.65171
	72	25	20	1.56959
	144	25	20	1.40336
	288	25	20	1.43327
39	40	25	5	1.53754
	78	25	5	1.38034
	156	25	5	1.60557
	312	25	5	1.44018
43	44	25	1	1.43727
	86	25	1	1.92402
	172	25	1	1.56497
	344	25	1	1.59893
46	46	10	1	1.55024
	92	10	1	1.17946
	184	10	1	1.78154
	368	10	1	1.15330

Pollard's method generally results in $t/\sqrt{p} \approx 1.8$, while Teske [56] presents an average of $t/\sqrt{p} \approx 1.45$. From the table, we see that, when $s = 4b$, our results seem to be near $t/\sqrt{p} \approx 1.49$ (with larger fluctuation when $M_2 \leq 5$), a result comparable to Teske's.

Chapter 3

Identity-Based Encryption from Bilinear Maps

3.1 Introduction

Shamir asked for an identity-based encryption (IBE) cryptosystem in 1984 [51], but a fully functional IBE scheme was not found until recent work by Boneh and Franklin [14] and Cocks [19]. Recall that an IBE scheme is a public-key cryptosystem where any arbitrary string is a valid public key. The corresponding private keys must be computed by a trusted third party called the private key generator (PKG) (who possesses a master secret). Users of the system request their private key from the PKG.

We note that the public key infrastructure associated with standard public-key cryptosystems also includes a trusted third party (in the form of a root certificate authority) and allows a hierarchy of certificate authorities [65]: the root certificate authority can issue certificates for other certificate authorities, who in turn can issue certificates for users in their respective domains.

The original system of Boneh and Franklin does not allow for such structure. However, a hierarchy of PKGs is desirable in an IBE system, as it greatly reduces the workload on the master server(s) and allows key escrow at several levels. For instance, if the users of the system are employees of corporations, then it is natural

to want each corporation to be able to generate the private keys for their employees, so that employees request their keys from their corporation, rather than the top-level PKG. Only corporations make requests of the top-level PKG (each corporation will make one request, for their domain secret). This is the idea behind a hierarchical IBE (HIBE) system. In particular, this is an example of a two-level HIBE (2-HIBE) scheme. (The main advantage of an HIBE system over standard PKI is that senders can derive the recipient's public key from their address without an online lookup.)

More precisely, there are three types of entities in a 2-HIBE scheme. There is the root PKG, who possesses a master key. In the upper level, there are domain PKGs, who can request their domain key from the root PKG. Lastly, there are users, who can request private keys from their domain PKG. Each user and each domain has a primitive ID (PID), which is an arbitrary string. (If Alice works for Company.com and her email address is `alice@company.com`, her PID is `alice` and her company's PID is `company.com`.) The public key of a user consists of a tuple of PIDs: the PID of the user and the PID of the user's domain (this public key is also called the user's address) and, as with IBE systems, it is clear that a sender can derive the receiver's public key offline. We can generalize to HIBE schemes with more levels by allowing subdomains, subsubdomains, and so on.

Another application for HIBE systems is generating short-lived keys for portable computing devices. Suppose Alice is planning to embark on a week-long business trip and wants to be able read her encrypted mail while on the road. However, she is also worried that her laptop may be stolen or otherwise compromised, so she does not want to simply copy her private key to the laptop. This dilemma is readily solved with a 2-HIBE system: this time, the upper level consists of people, such as Alice, and the lower level consists of dates, and when an arbitrary user (say, Bob) wants to send a message to Alice he uses the tuple of Alice's PID and the PID for the current date as her address. Alice can generate (for example) seven days' worth of keys (from her private key that she has previously requested from a PKG) and transfer these to her laptop. Now if the laptop is compromised, the damage is limited. We note that collusion at the bottom level is not an issue, as Alice will only put a small number of keys on her laptop. This problem can also be solved with a standard (nonhierarchical)

IBE scheme by having Alice run her own IBE system [14], but in this case Bob must get Alice's system parameters before he can communicate with her.

In Section 3.2, we give formal security definitions that can model plausible real-life attack scenarios on HIBE systems. In addition to chosen-ciphertext attacks, we must also worry about attacks involving collusion by entities on arbitrary levels. In our example above, for instance, if the domain PKG of one corporation A colludes with employees of another corporation B , they should not be able to decrypt messages of other employees of corporation B (or of any other corporation C , for that matter). In general, an adversary should not be able to decrypt a message encrypted for a particular user in a particular domain (and subdomain, subsubdomain, etc.), even if they have access to the private key of every other user and of every other domain (and subdomain, subsubdomain, etc.), in addition to information obtained from a decryption oracle.

In Section 3.3, we present a 2-HIBE scheme with total collusion resistance at the upper level and partial collusion resistance at the lower level. (This limitation does not affect its applicability to the above laptop example.) In terms of the corporate setting, even if an arbitrary number of corporations collude, the master secret is safe, but, at the lower level, if more than a certain number of employees of a corporation C collude, they can expose C 's private key. In Section 3.4, we describe an HIBE scheme constructed by Gentry and Silverberg [29, 30] that achieves total collusion resistance at all levels.

Both systems require a bilinear map with certain properties. A suitable map can be constructed from the Weil pairing (which is described in [14]). Its performance is sufficiently fast for practical purposes, provided the number of colluding parties allowed in the lower level is not too large. (Its running time and key size involve a term linear in this number.) Additionally, we can employ the same techniques used with the Boneh-Franklin IBE scheme to split secrets across several servers and achieve robustness for free.

3.2 Definitions

An identity-based encryption scheme (IBE) is specified by four randomized algorithms: **Setup**, **KeyGen** (called **Extract** in [14]), **Encrypt**, and **Decrypt**. In brief, **Setup** generates system parameters that are publicly released and a master key that is given to the PKG only; **KeyGen** is run by the PKG to generate private keys corresponding to a given primitive ID (PID); **Encrypt** encrypts a message using a given PID (PIDs are public keys); and **Decrypt** decrypts a ciphertext given a private key. We shall always take the message space to be $\mathcal{M} = \{0, 1\}^m$.

These algorithms must satisfy the standard consistency constraint, namely, when d is the private key generated by algorithm **KeyGen** when it is given the PID A as the public key, then

$$\forall M \in \mathcal{M} : \text{Decrypt}(\text{params}, A, C, d) = M ,$$

where $C = \text{Encrypt}(\text{params}, A, M)$.

An ℓ -HIBE scheme has a family of ℓ key-generation algorithms (**KeyGen** $_i$ for $1 \leq i \leq \ell$) instead of just one, and public keys are now ℓ -tuples of PIDs instead of just a single PID.

Definition 3.2.1 A primitive ID (PID) is an arbitrary string, i.e., an element of $\{0, 1\}^*$.

Definition 3.2.2 An address is an ℓ -tuple of PIDs.

An address fully specifies a user's public key.

Definition 3.2.3 A prefix address (or prefix) is an i -tuple of PIDs for some $0 \leq i \leq \ell$. A prefix address $\langle S_1, \dots, S_i \rangle$ is said to be a prefix of the prefix address $\langle T_1, \dots, T_j \rangle$ if $i \leq j$ and $S_a = T_a$ for $1 \leq a \leq i$.

Notice that addresses also happen to be prefix addresses.

Definition 3.2.4 For a nonnegative integer ℓ , an ℓ -level hierarchical identity-based encryption scheme (ℓ -HIBE) scheme is specified by $\ell+3$ randomized algorithms: **Setup**, **KeyGen** $_i$ (for $1 \leq i \leq \ell$), **Encrypt**, and **Decrypt**:

Setup: *Input:* security parameter $k \in \mathbb{N}$. *Output:* system parameters **params** and a master key mk_e (which we also call the level-0 key).

KeyGen_i (for $1 \leq i \leq \ell$): *Input:* **params**, $\text{mk}_{\langle S_1, \dots, S_{i-1} \rangle}$ (a level- $(i-1)$ key), and an i -tuple of PIDs (a prefix address). *Output:* $\text{mk}_{\langle S_1, \dots, S_i \rangle}$ (a level- i key).

Encrypt: *Input:* **params**, an address, and a message. *Output:* a ciphertext.

Decrypt: *Input:* **params**, an address, a ciphertext, and a private key $\text{mk}_{\langle S_1, \dots, S_\ell \rangle}$. *Output:* the corresponding plaintext.

These algorithms must satisfy the standard consistency constraint, namely, if $\text{mk}_{\langle S_1, \dots, S_\ell \rangle}$ is the private key generated by algorithm **KeyGen_ℓ** when it is given the address $\langle S_1, \dots, S_\ell \rangle$ as the public key, then

$$\forall M \in \mathcal{M} : \text{Decrypt}(\text{params}, \langle S_1, \dots, S_\ell \rangle, C, \text{mk}_{\langle S_1, \dots, S_\ell \rangle}) = M ,$$

where $C = \text{Encrypt}(\text{params}, \langle S_1, \dots, S_\ell \rangle, M)$.

We note that any user with the key for a prefix address $\langle S_1, \dots, S_i \rangle$ (e.g., the employer of the user described by $\langle S_1, \dots, S_\ell \rangle$) will be able to decrypt messages encrypted for $\langle S_1, \dots, S_\ell \rangle$ because they have the ability to generate the key $\text{mk}_{\langle S_1, \dots, S_\ell \rangle}$.

Remark 3.2.1 *For certain values of ℓ , an HIBE scheme is the same as other familiar structures:*

- When $\ell = 0$, this definition captures the essence of a public-key encryption scheme: the level-0 key corresponds to a private key and the **params** correspond to the public key (the address is empty when calling **Encrypt**; each system is associated with only one private key/public key pair).
- When $\ell = 1$, we have a definition of a standard IBE.

3.2.1 Security

In order to cover realistic attacks, we assume that an attacker may be able to obtain private keys at any level except for the master secret, and extend the standard model of chosen-ciphertext security accordingly. We note that if the master secret is compromised, the effects are at least as disastrous as when the root certificate authority is compromised in a public-key cryptosystem. Thus we assume that the precautions taken to guard the master secret are similar to those taken to guard a root certificate authority in real life (*e.g.*, secret splitting, tamper-resistant hardware), rendering it unassailable. Consider the following game played by two parties, an adversary and a challenger:

1. The challenger runs the **Setup** algorithm (for a given security parameter k) and gives **params** to the adversary. It does not divulge \mathbf{mk}_ϵ .
2. The adversary submits any number of decryption and/or key-generation queries adaptively (*i.e.*, each query may depend on the replies to previous queries). In a decryption query, the adversary sends a ciphertext and an address and is given the corresponding plaintext under the unique key associated with that address (assuming the ciphertext and address are valid). For a key-generation query, the adversary submits any prefix address $\langle S_1, \dots, S_i \rangle$ (for some $1 \leq i \leq \ell$), and is told the output K_i (where K_j is defined to be $\mathbf{KeyGen}_j(K_{j-1}, \langle S_1, \dots, S_j \rangle)$ for $0 < j \leq i$ and K_0 is the key returned by **Setup**).

In other words, not only can the adversary learn the decryption of any chosen ciphertext, it can also obtain the key corresponding to any prefix address.

3. The adversary then outputs any two plaintexts $M_0, M_1 \in \mathcal{M}$ and any address N on which it wishes to be challenged, subject to the restriction that no prefix of N has been queried in the previous step.
4. The challenger picks $b \in \{0, 1\}$ randomly and computes the ciphertext $C = \mathbf{Encrypt}(\mathbf{params}, N, M_b)$. It then sends the challenge C to the adversary.

5. The adversary again issues any number of decryption and/or key-generation queries adaptively, except that it now may not ask for the key corresponding to any prefix of N or for the plaintext corresponding to C under the private key corresponding to N .
6. The adversary outputs $b' \in \{0, 1\}$, and wins if $b = b'$.

We call such an adversary an ID-CCA attacker.

Definition 3.2.5 *We define an HIBE scheme to be secure against adaptive chosen-ciphertext attack (ID-CCA) if no polynomially bounded adversary has a non-negligible advantage in the above game, that is, for any polynomial f and for any probabilistic polynomial-time algorithm \mathcal{A} , $\text{Adv}(\mathcal{A}) := |\Pr[b = b'] - \frac{1}{2}|$ is less than $1/f(k)$. (The probability is over the random bits used by the two parties.)*

In Section 3.3, we describe a 2-HIBE scheme that is secure provided the adversary is limited to n KeyGen_2 queries within its domain (for a given n ; unlimited KeyGen_1 queries are allowed). In other words, our system resists arbitrary collusion at the domain level, but resists only limited collusion at the user level. In Section 3.4, we see the Gentry-Silverberg system, which is secure against unlimited collusion.

We will also utilize a weaker notion of security in intermediate steps of our proofs. Consider another game played by two parties, an adversary and a challenger:

1. The challenger runs the **Setup** algorithm (for a given security parameter k) and gives **params** to the adversary.
2. The adversary submits some number of key-generation queries adaptively, that is, for each query, the adversary submits any prefix address $\langle S_1, \dots, S_i \rangle$ (for some $1 \leq i \leq \ell$), and is told the output K_i (where K_j is defined to be $\text{KeyGen}_j(K_{j-1}, \langle S_1, \dots, S_j \rangle)$ for $0 < j \leq i$ and K_0 is the key returned by **Setup**).
3. The adversary then outputs any address N on which it wishes to be challenged, subject to the restriction that no prefix of N has been queried in the previous step.

4. The challenger picks a message $M \in \mathcal{M}$ randomly and computes the associated ciphertext, $C = \text{Encrypt}(\text{params}, N, M)$. It sends the challenge C to the adversary.
5. The adversary again issues some number of key-generation queries adaptively, except that it now may not ask for the key corresponding to any prefix of N .
6. The adversary outputs some message $M' \in \mathcal{M}$, and wins if $M = M'$.

We call such an adversary an ID-OWE attacker.

Definition 3.2.6 *We define an HIBE scheme to be a one-way identity-based encryption scheme (ID-OWE) if no polynomially bounded adversary has a non-negligible advantage in the above game.*

Both these definitions are generalizations of definitions originally given by Boneh and Franklin [14].

3.3 An HIBE Scheme Resistant against Domain Collusion

We present a two-level system resistant to collusion at the domain level. The system is based on bilinear forms between two prime-order groups.

3.3.1 The BDH Assumption

We briefly review definitions given by Boneh and Franklin [14, 15].

Definition 3.3.1 *Let G_1, G_2 be groups with prime order q . Then we say a map $e: G_1 \times G_1 \rightarrow G_2$ is bilinear if, for all $g, h \in G_1$ and $a, b \in \mathbb{F}_q$, we have $e(g^a, h^b) = e(g, h)^{ab}$.*

Definition 3.3.2 *The Bilinear-Diffie-Hellman problem (BDH) for a bilinear function $e: G_1 \times G_1 \rightarrow G_2$ such that $|G_1| = |G_2| = q$ is prime is defined as follows: given $g, g^a, g^b, g^c \in G_1$, compute $e(g, g)^{abc}$, where g is a generator and a, b, c are randomly chosen from \mathbb{F}_q . An algorithm is said to solve the BDH problem with an advantage of ε if*

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \geq \varepsilon .$$

Definition 3.3.3 *A randomized algorithm \mathcal{IG} that takes as input a security parameter $k \in \mathbb{N}$ (in unary) is a BDH parameter generator if it runs in time polynomial in k and outputs the description of two groups G_1, G_2 and a bilinear function $e: G_1 \times G_1 \rightarrow G_2$. We further require that the groups have prime order (which we call q), and denote the output of the algorithm by $(G_1, G_2, e) = \mathcal{IG}(1^k)$.*

Definition 3.3.4 *We say that \mathcal{IG} satisfies the BDH assumption if no probabilistic polynomial-time algorithm \mathcal{A} can solve BDH (for $\mathcal{IG}(1^k)$) with non-negligible advantage.*

Henceforth, we make use of some fixed BDH parameter generator \mathcal{IG} that satisfies the BDH assumption and use the symbols G_1, G_2, e, q to represent the constituents of its output. Boneh and Franklin [14] also give details on how to implement such a generator (their system also required one), based on the Weil pairing. (In their construction, G_1 is a group of points on a certain elliptic curve and G_2 is a certain subgroup of $\mathbb{F}_{p^2}^\times$, for some prime p .)

This assumption was implicitly used by Joux [34] to build a one-round three-party Diffie-Hellman protocol. Other constructions also require the BDH assumption ([35, 62, 63]). Additionally, a bilinear function is needed in a recently described short signature scheme [16].

3.3.2 A Game Transformation

The BDH assumption is closely tied to the CDH assumption. Recall that the CDH problem asks for g^{ab} given g, g^a, g^b , whereas the goal in the CDH problem is to compute

$e(g, g)^{abc} = e(g^{ab}, g^c)$ given g^c in addition to g, g^a, g^b . This similarity between the BDH and CDH assumptions naturally leads to the following transformation on games:

Definition 3.3.5 *Using the notation of the previous section, suppose \mathcal{G} is a game where the goal of the adversary is to compute a particular element $g \in G_1$. Then the e -transformation of \mathcal{G} is the same game as \mathcal{G} except now the adversary is also given a random $h \in G_1$ and the adversary's goal is to compute $e(g, h)$.*

We can transform assumptions by applying this transformation to the underlying game. For example, we obtain the BDH assumption (associated with a particular e) when we apply this transformation to the CDH assumption.

It is possible to formulate our assumptions differently: we could have started with assuming that e is a bilinear function such that if a game \mathcal{G} is hard then its e -transformation is also hard. This would simplify our exposition (for example, we need only assume the CDH problem is hard, as that implies that the BDH problem is hard). However, such an assumption is really an abstract description of a class of assumptions, and we prefer the readability gained by relying on a small number of concrete assumptions instead.

Clearly, if an adversary can win a game \mathcal{G} , then it can easily win the e -transformation of \mathcal{G} . The converse is far from clear.

We shall see that transformed assumptions are required to show that schemes are ID-OWE; without transformation, the assumptions are more natural, but we can only show that an adversary cannot recover a user's private key.

3.3.3 Linear e -One-Way Functions

We now build up to the definition of a linear e -one-way function, from which one could build an HIBE scheme. We then construct a function that is weaker than e -one-way that will allow for efficiently building a 2-HIBE scheme that is secure against any collusion at the domain level and limited collusion at the user level.

Suppose that we have a function $h: G \times X \rightarrow G_1$, where G and G_1 are groups, G is of prime order p , X is a set, and $h(g^a, x) = h(g, x)^a$ for all $g \in G, x \in X, a \in \mathbb{F}_p$.

Definition 3.3.6 *The elements $x, x_1, x_2, \dots, x_n \in X$, $a \in \mathbb{F}_p$, and a generator $g \in G$ are chosen at random. Given x, g , and $\langle x_i, h(g^a, x_i) \rangle$ for $i = 1, 2, \dots, n$, the problem of computing $h(g^a, x)$ is called the linear one-way problem (of size n).*

Definition 3.3.7 *We say that h is a linear one-way function if no probabilistic polynomial-time (in n and $\log p$) algorithm can solve the linear one-way problem of any size.*

Remark 3.3.1 *For example, if DDH is hard in \mathbb{F}_p^\times , the Weil pairing is an example of a linear one-way function. More generally, bilinear functions that satisfy BDH give rise to families of linear one-way functions. For example, suppose we have $(G_1, G_2, e) = \mathcal{IG}(1^k)$. Then fix a generator $g \in G_1$ and consider the function $f_g: G_1 \rightarrow G_2$ defined by $f_g(g_1) := e(g, g_1)$. Now, f_g is one-way, assuming DDH is hard in G_2 . To see this, assume that f_g is easy to invert; DDH in G_2 can be solved as follows: given $x, x^a, x^b, x^c \in G_2$ we find their inverses y, y_a, y_b, y_c respectively, and check if $e(y, y_c) = e(y_a, y_b)$. We note that if \mathcal{IG} is constructed as described by Boneh and Franklin, then G_2 is a subgroup of \mathbb{F}_p^\times , a group in which DDH is thought to be hard. (It is also possible to construct elliptic curves where the q -torsion points are contained in \mathbb{F}_p for some large prime q . Inverting the Weil pairing on these curves is equivalent to breaking DDH in \mathbb{F}_p .) More generally, this is why the relationship between a game and its e -transformation appears to be highly nontrivial: if an algorithm \mathcal{A} could win a game \mathcal{G} , given an algorithm \mathcal{B} that wins the e -transformation of \mathcal{G} , then \mathcal{A} is an algorithm that can invert f_g .*

Now suppose that $(G_1, G_2, e) = \mathcal{IG}(1^k)$. Then the e -transformation of the linear one-way problem is called the linear e -one-way problem. (In this problem, we are also given g^r for some random $r \in \mathbb{F}_p$ (in addition to x, g , and $\langle x_i, h(g^a, x_i) \rangle$ for $i = 1, 2, \dots, n$) and now the goal is to compute $e(h(g^a, x), g^r)$.)

Definition 3.3.8 *If no probabilistic polynomial-time algorithm can solve the linear e -one-way problem of any size, then we say that h is a linear e -one-way function.*

If we knew how to construct linear e -one-way functions, we could construct a 2-HIBE scheme as follows:

Setup: Input: $k \in \mathbb{N}$. Run $\mathcal{IG}(1^k)$ and set (G_1, G_2, e) to be the output. Construct a linear e -one-way function $h: G_1 \times \mathbb{F}_q \rightarrow G_1$. Choose a random $a \in \mathbb{F}_q$ and a random generator $g \in G_1$. Pick cryptographically strong hash functions $H_1: \{0, 1\}^* \rightarrow G_1$, $H_2: \{0, 1\}^* \rightarrow \mathbb{F}_q$, and $H_3: G_2 \rightarrow \{0, 1\}^m$ (for some m).

Output: $\text{mk}_\epsilon := a$, and $\text{params} := \langle G_1, G_2, e, g, g^a, H_1, H_2, H_3 \rangle$.

KeyGen₁: Input: a prefix address $\langle S \rangle$ (the domain name).

Output: $\text{mk}_{\langle S \rangle} := H_1(S)^a \in G_1$.

KeyGen₂: Input: an address $\langle S, T \rangle$ (S is the domain PID and T is the user PID).

Let $\text{mk}_{\langle S \rangle} \in G_1$ be the domain key.

Output: $k = \text{mk}_{\langle S, T \rangle} := h(\text{mk}_{\langle S \rangle}, H_2(S \| T)) \in G_2$.

Encrypt: Input: params , $N = \langle S, T \rangle$ (S is the recipient domain's PID and T is the recipient user's PID), and M .

Pick a random $r \in \mathbb{F}_q$.

Output: $C = \langle g^r, M \oplus H_3(s) \rangle$, where $s := e(h(H_1(S), H_2(S \| T)), g^a)^r$.

Decrypt: Input: params , $N = \langle S, T \rangle$, a ciphertext $C = \langle U, V \rangle$, and a user's private key $k := \text{mk}_{\langle S, T \rangle} \in G_2$.

Output: $M = V \oplus H_3(e(k, U))$.

It can be shown that this scheme is ID-OWE. By applying the Fujisaki-Okamoto transformation [28], we obtain a scheme which is ID-CCA. Though finding a linear e -one-way function h remains an open problem, we are able to construct an h such that the linear e -one-way problem for a fixed n is hard, giving rise to a 2-HIBE system that is resistant to (unlimited) domain-level collusion and can tolerate up to n -party user-level collusion. We describe this in the following section. Briefly, we will define $h: G_1^{n+1} \times \mathbb{F}_q \rightarrow G_1$ (for some n ; q is the prime order of G_1) as $h((g_0, g_1, \dots, g_n), d) := g_0^{d^0} g_1^{d^1} \cdots g_n^{d^n}$. We then have a linear function h such that,

given g and n pairs $\langle x_i, h(g, x_i) \rangle$, it appears hard to determine $\langle x', h(g, x') \rangle$ for any other x' .

3.3.4 Our Domain-Collusion Resistant Scheme

Let n denote the amount of collusion that we are willing to tolerate at the user level.

Setup: Input: $k \in \mathbb{N}$. Run $\mathcal{IG}(1^k)$ and set (G_1, G_2, e) to be the output. Choose a random $a \in \mathbb{F}_q$ and a random $g \in G_1$. Pick cryptographically strong hash functions $H_1: \{0, 1\}^* \rightarrow G_1^{m+1}$, $H_2: \{0, 1\}^* \rightarrow \mathbb{F}_q$, and $H_3: G_2 \rightarrow \{0, 1\}^m$ (where $\mathcal{M} = \{0, 1\}^m$ is the message space). For the security proof, we view the hash functions as random oracles.

Output: $\text{mk}_\epsilon := a$ and $\text{params} := \langle G_1, G_2, e, g, g^a, H_1, H_2, H_3 \rangle$.

KeyGen₁: Input: a prefix address $\langle S \rangle$ (the domain name). Let $\langle g_0, g_1, \dots, g_n \rangle = H_1(S)$ (so each g_i lies in G_1).

Output: $\text{mk}_{\langle S \rangle} := \langle g_0^a, g_1^a, \dots, g_n^a \rangle \in G_1^{m+1}$.

KeyGen₂: Input: an address $\langle S, T \rangle$ (S is the domain PID and T is the user PID).

Set $d := H_2(S \| T)$ (which is an element of \mathbb{F}_q).

Let $\text{mk}_{\langle S \rangle} = \langle g_0^a, g_1^a, \dots, g_n^a \rangle \in G_1^{m+1}$ be the domain key.

Output: $k = \text{mk}_{\langle S, T \rangle} := \prod_{i=0}^n g_i^{ad^i} \in G_1$.

Encrypt: Input: params , $N = \langle S, T \rangle$ (S is the recipient domain's PID and T is the recipient user's PID), and a message M .

Set $\langle g_0, g_1, \dots, g_n \rangle := H_1(S)$. Set $d := H_2(S \| T)$.

Pick a random $r \in \mathbb{F}_q$. Then compute $w := e\left(\prod_{i=0}^n g_i^{d^i}, g^a\right)^r \in G_2$.

Output: the ciphertext $\langle g^r, M \oplus H_3(w) \rangle$.

Decrypt: Input: params , $N = \langle S, T \rangle$, a ciphertext $C = \langle U, V \rangle$, and a private key

$$k = \text{mk}_{\langle S, T \rangle} \in G_1.$$

$$\text{Output: } M = V \oplus H_3(e(k, U)).$$

The scheme is consistent because, when $U = g^r$, the bilinearity of e tells us that $e(k, U) = e\left(\prod_{i=0}^n g_i^{d_i}, g^a\right)^r$.

3.3.5 Proof of Security

Recall that we are restricting the adversary to at most n KeyGen_2 queries from the same domain.

Theorem 3.3.1 *Suppose \mathcal{A} is an ID-OWE attacker of our scheme with an advantage of ε . Then, if we model H_1 , H_2 , and H_3 as random oracles, there exists an algorithm \mathcal{B} that can solve the BDH problem with an advantage of $\varepsilon / (2(Q_{K_1} + 2Q_{K_2})Q_{H_1} \binom{Q_{H_2}}{n} \mathbf{e})$, where Q_{K_i} is the total number of KeyGen_i queries, Q_{H_i} is the number of H_i queries issued by \mathcal{A} , and \mathbf{e} is the base of the natural logarithm.*

Proof. The proof of the theorem is broken into several lemmata. In Lemma 3.3.2, we show that an attacker \mathcal{B} , whose KeyGen queries are restricted to only KeyGen_2 queries from the same domain as the challenge address, is essentially as strong as an arbitrary attacker \mathcal{A} . We do so in a manner similar to that used in the analysis of the Boneh-Franklin scheme [14], which is itself partly based on a technique of Coron [22]. In Lemma 3.3.3, we define the Bilinear Polynomial Diffie-Hellman (BPDH) game, and give a reduction from the attack by the \mathcal{B} described above to an attack by (an attacker) \mathcal{C} on the BPDH game. Lastly, Lemma 3.3.4 gives a reduction from an attack by \mathcal{C} on the BPDH game to an attack by \mathcal{D} on the BDH problem. The combination of the three lemmata leads immediately to the theorem. \square

Lemma 3.3.2 *Suppose there exists an ID-OWE attacker \mathcal{A} with an advantage of ε . Let Q_i be a bound on the number of H_i queries made by \mathcal{A} (for $i = 1, 2$). If we model H_1 as a random oracle, then there exists an ID-OWE attacker \mathcal{B} with an advantage of $\varepsilon / \mathbf{e}Q$, where $Q = Q_1 + 2Q_2$, whose key-generation queries are all KeyGen_2 queries from the same domain as the challenge domain.*

Proof. After receiving the system parameters, \mathcal{B} passes them on to \mathcal{A} . Without loss of generality, we may assume that every key-generation query for a prefix address (domain name) $\langle S \rangle$ or address $\langle S, T \rangle$ has been preceded by an H_1 (domain-level) hash query on the domain PID S . We may also assume that \mathcal{A} issues an H_1 hash query on the challenge domain PID before revealing it.

We will need some auxiliary functions and global variables:

Initially, L is an empty list that will hold information on \mathcal{B} 's responses to H_1 queries, and $s_{\text{CHALLENGE}}$ is a string that is set to a special value `NULL`. Additionally, we will use a unique value `REAL` (not in G, \mathbb{F}_q , etc.) in the proof.

When \mathcal{A} issues an H_2 query for an address $\langle S, T \rangle$ (i.e., a hash query on $S \| T$), \mathcal{B} returns $H_2(S \| T)$.

When \mathcal{A} issues an H_1 query on a domain PID S , \mathcal{B} runs the following algorithm:

1. If L contains a tuple whose first element is S , then
 - (a) If L contains $\langle S, r_0, r_1, \dots, r_n \rangle$, then return $\langle g^{r_0}, g^{r_1}, \dots, g^{r_n} \rangle$.
 - (b) If L contains $\langle S, \text{REAL} \rangle$, then return $H_1(S)$.
2. Otherwise, flip a coin that takes the value 1 with probability p and 0 otherwise (p will be determined later).
 - (a) If $\text{COIN} = 1$, then pick random $r_0, r_1, \dots, r_n \in \mathbb{F}_q$. Insert the tuple $\langle S, r_0, r_1, \dots, r_n \rangle$ into L , and return $\langle g^{r_0}, g^{r_1}, \dots, g^{r_n} \rangle$.
 - (b) Otherwise, $\text{COIN} = 0$. In this case, insert $\langle S, \text{REAL} \rangle$ into L and return $H_1(S)$.

Since we are modelling H_1 as a random oracle, \mathcal{A} cannot distinguish between this simulation and the real H_1 .

When \mathcal{A} issues a `KeyGen1` query on a prefix address (domain name) $\langle S \rangle$, \mathcal{B} runs the following algorithm:

By assumption, \mathcal{A} has already issued an H_1 query for S .

1. If $\langle S, r_0, r_1, \dots, r_n \rangle$ is on the list L , return $\langle g^{ar_0}, g^{ar_1}, \dots, g^{ar_n} \rangle$.

2. Otherwise, $\langle S, \text{REAL} \rangle$ appears on L : output FAILURE and halt.

When \mathcal{A} issues a KeyGen_2 query on an address $\langle S, T \rangle$, \mathcal{B} runs the following algorithm:

Again by assumption, a hash query on S has already been issued. Let $d = H_2(S \| T)$.

1. If L contains $\langle S, r_0, r_1, \dots, r_n \rangle$, return $\langle g^{ar_0d^0}, g^{ar_1d^1}, \dots, g^{ar_nd^n} \rangle$.
2. Otherwise, $\langle S, \text{REAL} \rangle \in L$:
 - (a) If $s_{\text{CHALLENGE}} = S$, then \mathcal{B} issues the KeyGen_2 query (recall that \mathcal{B} is allowed to do this for the challenge domain).
 - (b) If $s_{\text{CHALLENGE}} \neq \text{NULL}$ then \mathcal{B} outputs FAILURE and halts.
 - (c) Otherwise, \mathcal{B} sets $s_{\text{CHALLENGE}} := S$ and issues the KeyGen_2 query.

Eventually, \mathcal{A} outputs a challenge address $\langle S, T \rangle$. If $\langle S, \text{REAL} \rangle \notin L$, then output FAILURE. If $\langle S, \text{REAL} \rangle \in L$ and $s_{\text{CHALLENGE}} \neq \text{NULL}$ and $s_{\text{CHALLENGE}} \neq S$, then output FAILURE. Otherwise (when $\langle S, \text{REAL} \rangle \in L$ and ($s_{\text{CHALLENGE}} = \text{NULL}$ or $s_{\text{CHALLENGE}} = S$)), set $s_{\text{CHALLENGE}} := S$.

The next round of queries is handled in the same manner as in the first round.

Finally, \mathcal{A} will output a guess M and halt; then \mathcal{B} outputs M and halts. Clearly, if \mathcal{A} is successful, then so is \mathcal{B} .

Recall that Q_1 is the number of KeyGen_1 queries. Then the probability that FAILURE is not output during such a query is at least p^{Q_1} (it is sufficient to have $\text{COIN} = 1$ for each query).

Recall that Q_2 is the number of KeyGen_2 queries. In the worst case, for every KeyGen_2 query on an address $\langle S, T \rangle$, L contains $\langle S, \text{REAL} \rangle$, and, once $s_{\text{CHALLENGE}}$ has been set, any other value for S will cause failure. So the probability that failure is not output during KeyGen_2 queries is bounded from below by p^{Q_2-1} .

After \mathcal{A} outputs the challenge address, the probability that $\langle S, \text{REAL} \rangle$ is on the list L is $1 - p$, and the probability $s_{\text{CHALLENGE}} = S$ or $s_{\text{CHALLENGE}} = \text{NULL}$ is at least p^{Q_2} . (In the worst case, every KeyGen_2 query is in a different domain, and, trivially, the

probability that $s_{\text{CHALLENGE}} = S$ or $s_{\text{CHALLENGE}} = \text{NULL}$ is no less than the probability that $s_{\text{CHALLENGE}}$ remains NULL.)

Let $k = Q_1 + 2Q_2 - 1$. Then $p^k(1-p)$ is a lower bound on the probability that a failure state is not reached. It is minimized when $p = k/(k+1)$, which makes the probability of not reaching a failure state bounded from below by $1/e(k+1)$.

With $Q = Q_1 + 2Q_2$, we see that \mathcal{B} has an advantage of at least ε/eQ . \square

Definition 3.3.9 *The Computational Polynomial Diffie-Hellman (CPDH) game (of degree n) for a function $H: X \rightarrow G_1$, where X is a set, is the following game:*

A polynomial $f(x) = c_0 + c_1x + \dots + c_nx^n$ with coefficients in \mathbb{F}_q is chosen at random. An element a is chosen at random from \mathbb{F}_q .

The attacker is given $g, g^a, g^{c_0}, g^{c_1}, \dots, g^{c_n}$ and $d \in \mathbb{F}_q$.

Then the attacker picks any $s \in X$, and learns $g^{af(H(s))}$. This step is repeated up to n times. (The attack may be adaptive.)

Lastly, the attacker wins if it can output the value of $g^{af(d)}$.

Remark 3.3.2 *For $n = 0$, this reduces to the CDH problem. (The adversary is not allowed to make any queries.)*

Definition 3.3.10 *The Bilinear Polynomial Diffie-Hellman (BPDH) game (of degree n) for a function $H: X \rightarrow G_1$ is the e -transformation of the corresponding CPDH game, i.e., it is the same as the previous game except that the attacker is also given g^r for some random $r \in \mathbb{F}_q$ and now the attacker's goal is to compute $e(g, g)^{arf(d)}$.*

Remark 3.3.3 *For $n = 0$ this reduces to the BDH problem.*

Lemma 3.3.3 *Suppose there exists an ID-OWE attacker \mathcal{B} with an advantage of ε whose key-generation queries are always KeyGen_2 queries of addresses from the same domain as the challenge domain, and furthermore, \mathcal{B} makes at most n such queries. (\mathcal{B} makes no KeyGen_1 queries.) Then, there exists an attacker \mathcal{C} that can win the BPDH game for H_2 with an advantage of $\varepsilon/(2Q)$, where Q is a bound on the number of H_1 queries that \mathcal{B} makes.*

Proof. Again we may assume that any key-generation query for an address is preceded by a hash query for that address, and that before the challenge address is output, a hash query for the challenge address will have been issued. We may also assume that each query (for any hash function) is distinct (since previous results can simply be cached). Also, without loss of generality, we may assume that \mathcal{B} makes *exactly* n KeyGen_2 queries.

The algorithm \mathcal{C} is given as input $g, g^a, g^{c_0}, g^{c_1}, \dots, g^{c_n}, g^r, d$ (using the notation employed in the description of the BPDH game; its goal is to compute $e(g, g)^{\text{arf}(d)}$). \mathcal{C} begins by giving \mathcal{B} the system parameters g, g^a .

There is a list L that is used to store H_3 queries and is initially empty.

\mathcal{C} picks a random i between 1 and Q . On the i th H_1 query for S (that \mathcal{B} makes), \mathcal{C} sets $s_{\text{CHALLENGE}} := S$ and returns $\langle g^{c_0}, g^{c_1}, \dots, g^{c_n} \rangle$. For all other H_1 queries, \mathcal{C} returns $H_1(S)$. Since we are modelling H_1 as a random oracle, the algorithm \mathcal{B} cannot distinguish between this simulation of H_1 and the real H_1 .

When \mathcal{B} issues an H_2 query for $\langle S, T \rangle$, \mathcal{C} returns $H_2(S \| T)$.

When \mathcal{B} issues an H_3 query for $s \in G_2$, \mathcal{C} returns $H_3(s)$, and inserts $\langle s, H_3(s) \rangle$ into L .

When \mathcal{B} issues a KeyGen_2 query on $\langle S, T \rangle$, if $S \neq s_{\text{CHALLENGE}}$, then \mathcal{C} outputs FAILURE and halts. Otherwise, \mathcal{C} issues a query for $g^{\text{arf}(H_2(S \| T))}$ and returns the result to \mathcal{B} .

Eventually, \mathcal{B} gives the challenge address $N = \langle S, T \rangle$ to \mathcal{C} . If $S \neq s_{\text{CHALLENGE}}$, then \mathcal{C} outputs FAILURE and halts. Otherwise, \mathcal{C} chooses a random $R \in \{0, 1\}^k$, and \mathcal{C} gives the ciphertext $C := \langle g^r, R \rangle$ to \mathcal{B} .

The next round of queries is handled in the same manner as in the first round.

Eventually, \mathcal{B} outputs its guess M and halts. Then, the algorithm \mathcal{C} looks for a tuple of the form $\langle s, M \oplus R \rangle$ in L ; if it cannot find it, \mathcal{C} outputs FAILURE. If \mathcal{B} is successful (*i.e.*, if $\text{Encrypt}(\text{params}, N, M) = C$), then $M = R \oplus H_3(s)$, where $s = e(g, g)^{\text{arf}(H_2(S \| T))}$. If \mathcal{C} finds the tuple in L , \mathcal{C} then knows $n + 1$ values of the function $x \mapsto e(g, g)^{\text{arf}(x)}$, so \mathcal{C} can compute $e(g, g)^{\text{arf}(d)}$ using Lagrange interpolation.

Notice that, since H_3 is a random oracle, if $\langle s, H_3(s) \rangle$ is not found in L , then the decryption of the ciphertext C is independent of the knowledge \mathcal{B} accumulated from

its various queries, which means that \mathcal{B} succeeds in this case with probability $1/2^k$.

The probability of success is $(1/Q)(1 - 1/2^k)$ (i must be guessed correctly), which is at least $1/(2Q)$. \square

Lemma 3.3.4 *Suppose there exists an algorithm \mathcal{C} that can win the BPDH game for a function $H: X \rightarrow G_1$ with an advantage of ε and suppose H may be modelled as a random oracle. Then there exists an attacker \mathcal{D} that can solve the BDH problem with an advantage of $\varepsilon/\binom{Q}{n}$, where Q is a bound on the number of H queries that \mathcal{C} makes.*

Proof. We may assume that $Q \geq n$, that all H queries are distinct (previous results can be cached), and that a query for $g^{af(H(s))}$ implies that \mathcal{C} has already issued a query for $H(s)$.

The algorithm \mathcal{D} is given g, g^x, g^y, g^z for randomly chosen $x, y, z \in \mathbb{F}_q$ (its goal is to compute $e(g, g)^{xyz}$). Set $y_0 := y$. There is a list L that holds responses to H queries that is initially empty. \mathcal{D} picks random $a_0, a_1, \dots, a_n, y_1, y_2, \dots, y_n \in \mathbb{F}_q$. \mathcal{D} then solves the system of equations

$$\begin{array}{rcccc} g_0^{a_0^0} \cdot g_1^{a_0^1} & \cdots & g_n^{a_0^n} & = & g^{y_0} \\ g_0^{a_1^0} \cdot g_1^{a_1^1} & \cdots & g_n^{a_1^n} & = & g^{y_1} \\ & & \vdots & & \\ g_0^{a_n^0} \cdot g_1^{a_n^1} & \cdots & g_n^{a_n^n} & = & g^{y_n} \end{array}$$

for the g_i . If we define the matrix A by $A_{ij} := a_i^j$, then, with high probability, the a_i are distinct (so A is a Vandermonde matrix), thus guaranteeing a unique solution for the g_i .

Then, \mathcal{D} hands \mathcal{C} the input $\langle g, g^x, g_0, g_1, \dots, g_n, g^z, a_0 \rangle$.

Let Q be a bound on the number of H queries made by \mathcal{C} . \mathcal{D} chooses a random subsequence I of length n from the sequence $(1, 2, \dots, Q)$.

Let $s_j \in X$ be the j th element on which \mathcal{C} makes an H query. \mathcal{D} answers that query as follows:

If j is the i th element of I , then \mathcal{D} responds with a_i and inserts $\langle s_j, i \rangle$ into L . Otherwise, $j \notin I$ and \mathcal{D} responds with a random number. Since the a_i were chosen

at random, the algorithm \mathcal{C} cannot distinguish between our simulation of a random oracle and a real random oracle.

If \mathcal{C} asks for $g^{af(H(s))}$, then, if $\langle s, i \rangle \in L$ for some i , \mathcal{D} replies with g^{y_i} . Otherwise, it outputs FAILURE and halts.

Eventually, \mathcal{C} will output its guess g' for $e(g, g)^{zzf(a_0)}$ and halt; then \mathcal{D} outputs g' and halts.

Let $r_1, r_2, \dots, r_n \in \mathbb{F}_q$ be such that $g_0 = g^{r_0}, g^{r_1}, \dots, g_n = g^{r_n}$ (we will never need to explicitly compute the r_i). If \mathcal{D} has not yet failed, and, if \mathcal{C} wins its game, then \mathcal{C} will output $g' = e(g, g)^{zzf(a_0)} = e(g, g)^{xyz}$ (where f is the polynomial $f(u) = r_0 + r_1u + \dots + r_nu^n$), which means that \mathcal{D} , by outputting $e(g, g)^{xyz}$, will win the BDH game.

Thus, the probability of \mathcal{D} succeeding, given that \mathcal{C} is successful, is at least $1/\binom{Q}{n}$ (it is sufficient for the set I to correspond exactly to the n elements of X for which \mathcal{C} issues $g^{af(H(s))}$ queries). \square

Applying the Fujisaki-Okamoto transformation to the cryptosystem yields an ID-CCA 2-HIBE system that tolerates user collusion up to size n . In this system, the Setup algorithm also selects two hash functions $H': \{0, 1\}^k \times \{0, 1\}^k \rightarrow \mathbb{F}_q$ and $H'': \{0, 1\}^k \rightarrow \{0, 1\}^k$, and there is an extra level of hashing during encryption and decryption, as follows (we use the same notation as in the description of the cryptosystem): **Encrypt** now also picks a random $\sigma \in \{0, 1\}^k$, computes $r := H'(\sigma, M)$ (instead of picking a random $r \in \mathbb{F}_q$), and outputs $C := \langle g^r, \sigma \oplus H_3(s), M \oplus H''(\sigma) \rangle$ (as opposed to $\langle g^r, M \oplus H_3(s) \rangle$). **Decrypt** is modified similarly.

The proof of Lemma 3.3.4 involves a reduction that is exponential in n , rendering the system untrustworthy for large n . In order to rectify this, one could simply assume that the Bilinear Polynomial Diffie-Hellman game is hard to win, but it is preferable to find a polynomial-time reduction from winning the Bilinear Polynomial Diffie-Hellman game to a more natural assumption.

3.4 The Gentry-Silverberg HIBE Scheme

In 2002, Gentry and Silverberg [29] extended the Horwitz-Lynn HIBE system to support total collusion-resistance at all levels. To follow up the discussion of the Horwitz-Lynn system, we present the details of the Gentry-Silverberg system here.

Gentry and Silverberg begin with a one-way identity-based encryption scheme (ID-OWE) and, as in the Boneh-Franklin and Horwitz-Lynn results, apply the Fujisaki-Okamoto transformation to build it to a chosen-ciphertext-secure scheme (ID-CCA). We describe the basic system here and then present its fundamental security result.

Setup: Input: $k \in \mathbb{N}$.

Run $\mathcal{IG}(1^k)$ and set (G_1, G_2, e) to be the output. Choose a random $a_0 \in \mathbb{F}_q$ and a random $g \in G_1$. Set $h_\epsilon := g^{a_0}$. Pick cryptographically-strong hash functions $H_1: \{0, 1\}^* \rightarrow G_1$ and $H_2: G_2 \rightarrow \{0, 1\}^m$ (where $\mathcal{M} = \{0, 1\}^m$ is the message space). For the security proof, we view the hash functions as random oracles. The ciphertext space will be $\mathcal{C} = (\cup_{i=1}^\ell G_1^i) \times \{0, 1\}^m$.

Output: $\text{mk}_\epsilon := a_0$ and $\text{params} := \langle G_1, G_2, e, g, g^{a_0}, H_1, H_2 \rangle$.

KeyGen_i: Input: params , an address $N = \langle S_1, \dots, S_i \rangle$, N 's parent's key $\text{mk}_{\langle S_1, \dots, S_{i-1} \rangle}$, and its parent's h -values $(h_{\langle S_1 \rangle}, \dots, h_{\langle S_1, \dots, S_{i-1} \rangle})$.

Pick a random $a \in \mathbb{F}_q$. Set $g_i := H_1(S_1 || \dots || S_i) \in G_1$. Set $\text{mk}_{\langle S_1, \dots, S_i \rangle} := g_i^a \text{mk}_{\langle S_1, \dots, S_{i-1} \rangle}$. Set $h_{\langle S_1, \dots, S_i \rangle} := g^a$.

Output: $k = \text{mk}_{\langle S_1, \dots, S_i \rangle}$ and the h -values $(h_{\langle S_1 \rangle}, \dots, h_{\langle S_1, \dots, S_i \rangle})$.

Encrypt: Input: params , $N = \langle S_1, \dots, S_\ell \rangle$, and a message M .

For $1 \leq i \leq \ell$, set $g_i := H_1(S_1 || \dots || S_i)$.

Pick a random $r \in \mathbb{F}_q$. Then compute $w := e(h_\epsilon, g_1)^r \in G_2$.

Output: the ciphertext $\langle g^r, g_2^r, g_3^r, \dots, g_\ell^r, M \oplus H_w(w) \rangle$.

Decrypt: Input: params , $N = \langle S_1, \dots, S_\ell \rangle$, N 's h -values $(h_{\langle S_1 \rangle}, \dots, h_{\langle S_1, \dots, S_\ell \rangle})$, a ciphertext $C = \langle U, U_2, U_3, \dots, U_\ell, V \rangle \in \mathcal{C}$, and a private key $k = \text{mk}_{\langle S_1, \dots, S_\ell \rangle} \in G_1$.

Output: $M = V \oplus H_2(e(U, k) / \prod_{i=2}^t e(h_{i-1}, U_i))$.

Gentry and Silverberg also present and prove the following fundamental security theorem, showing that their scheme is a one-way identity-based encryption scheme (ID-OWE):

Theorem 3.4.1 *Suppose \mathcal{A} is an ID-OWE attacker with advantage ε . Then, if we model H_1 and H_2 as random oracles, there exists an algorithm \mathcal{B} that can solve the BDH problem with an advantage of $\left(\varepsilon \left(\frac{\ell}{e^{(Q_K + \ell)}}\right)^\ell - \frac{1}{2^m}\right) / Q_{H_2}$ that runs in time $O(\text{TIME}(\mathcal{A}))$, where Q_{H_2} is the number of hash queries to H_2 , Q_K is the number private key extraction queries issues by \mathcal{A} , and e is the base of the natural logarithm.*

Chapter 4

Weak Trapdoors from r th-Power Residues

4.1 Introduction

Let $N = pq$ be an RSA modulus and let r divide $p - 1$ and $q - 1$. Let $\text{PR}_r(N)$ be the set of r th residues in \mathbb{Z}_N^\times , *i.e.*, $\text{PR}_r(N) = (\mathbb{Z}_N^\times)^r$. The r th-residuosity problem for a given N is the problem of distinguishing the uniform distribution on $\text{PR}_r(N)$ from the uniform distribution on \mathbb{Z}_N^\times . We refer to this problem as RESIDUE_r . The problem is believed to be hard unless one is given the factorization of N .

The intractability of RESIDUE_r is the basis of a number of cryptosystems [9, 10, 11, 21, 41]. For example, Benaloh [9] constructs an additively homomorphic encryption scheme based on RESIDUE_r as follows: let r be a product of small primes and let $N = pq$ where $r|(p - 1)$, $r^2 \nmid (p - 1)$, and $r \nmid (q - 1)$. Let $g \in \mathbb{Z}_N^\times$ be an element such that r divides the order of g . Then encryption of a message $m \in \mathbb{Z}_r$ is defined as: $\text{E}_{r,N,g}(m) = g^m h^r \pmod{N}$, where $h \in \mathbb{Z}_N^\times$ is picked at random (for each message). The system is semantically secure against a passive adversary, assuming that RESIDUE_r is hard. Decryption is done using the factorization of N as the secret key. This system is used to build anonymous voting systems and cryptographic counters. More efficient variants of this system were recently proposed by Okamoto-Uchiyama [42] and Pallier [43].

Following preliminary material in Sections 4.2 and 4.3, we present in Section 4.4 a system very similar to Benaloh's: we encrypt $m \in \mathbb{Z}_r$ by computing $E(m) = g^m h^r \pmod{N}$; here, however, r divides both $p-1$ and $q-1$. We show that decryption here does not require the factorization of N , but only $\mu \in \mathbb{Z}_N^\times$, a nontrivial root of unity (*i.e.*, $\mu^r = 1$, and μ has order r in both \mathbb{Z}_p^\times and \mathbb{Z}_q^\times). When r divides both $p-1$ and $q-1$, it is believed that such $\mu \in \mathbb{Z}_N^\times$ does not reveal the factorization of N . Therefore, decryption can be done using a trapdoor (μ) that is weaker than the factorization of N . We base semantic security of this system on the hardness of the r th-residuosity problem.

Since decryption in this system can be done without the factorization of N , the same N can be used for other tasks. For example, by giving a different $\mu \in \mathbb{Z}_N^\times$ to each recipient, we show how to construct a weak form of broadcast encryption: the sender (who has only N) can encrypt a message so that each recipient is only able to decrypt a prespecified fragment of the message. We also obtain a form of signcryption [64, 3]: suppose user Alice has the factorization of N , while user Bob only has $\mu \in \mathbb{Z}_N^\times$. Then Alice can use N to sign messages using RSA while Bob can decrypt messages encrypted modulo N . By combining these two facts, Alice can send messages to Bob which are both signed and encrypted.

We also present further evidence of the usefulness of the r th-power-residue symbol in Section 4.5. There, we show how to speed up the elliptic-curve method (ECM) [36] for factoring numbers of the form $N = pq^r$ (when $r > 2$). Previously, Okamoto and Peralta [44] showed how to speed up the large-prime variant of the ECM for numbers of the form $N = pq^2$. We generalize their technique using Eisenstein reciprocity and show how to speed up ECM for numbers of the form $N = pq^r$ for small r (*e.g.*, $r = 3, 4, 5, \dots$). The only other factoring algorithm specifically designed to factor such numbers [13] works only when r is relatively large, *i.e.*, $r \approx \sqrt{\log p}$. We note the growing interest in studying the security of $N = pq^r$ in light of the results of Takagi [55].

4.2 The r th-Power-Residue Symbol

This paper presents various results that make use of the r th-power-residue symbol. We first briefly review the necessary facts about the power-residue symbol (more details are available, for example, in Chapter 14 of [33]). We define the symbol as a generalization of the Jacobi symbol. Throughout this section and the rest of the paper we assume that r is an odd integer greater than 1.

Definition 4.2.1 *For any positive integers r and N , $x \in \mathbb{Z}_N^\times$ is said to be an r th residue modulo N if there exists a $y \in \mathbb{Z}_N^\times$ such that $y^r = x \pmod{N}$.*

We denote by $\text{PR}_r(N) \subseteq \mathbb{Z}_N^\times$ the set of r th residues modulo N .

Throughout the paper we assume that $N = pq$, where r divides both $p - 1$ and $q - 1$. To see why we only consider this case, observe that if r divides neither $p - 1$ nor $q - 1$, then all $x \in \mathbb{Z}_N^\times$ are r th residues and, thus, distinguishing residues from non-residues is trivial. When r divides only one of $p - 1$ or $q - 1$, an r th root of unity $\mu \neq 1$ would reveal the factorization of N (since $\gcd(N, \mu - 1) \in \{p, q\}$). As discussed in the next section, when r divides both $p - 1$ and $q - 1$, an r th root of unity does not reveal the factorization of N , as far as we know. Consequently, we always use $N = pq$ where r divides both $p - 1$ and $q - 1$.

Let $\zeta_r = e^{2\pi i/r} \in \mathbb{C}$ be a complex primitive r th root of unity. We denote by D_r the ring of integers in $\mathbb{Q}[\zeta_r]$. Before we define the general r th-power-residue symbol, we start with a special case. The definition below is analogous to the definition of the Legendre symbol $\left(\frac{x}{p}\right)$ for an integer x over an integer prime p .

Definition 4.2.2 *For $x \in D_r$ and P a prime ideal not containing r , the r th-power-residue symbol $\left(\frac{x}{P}\right)_r$ is defined so that*

$$\left(\frac{x}{P}\right)_r = \begin{cases} 0 & \text{if } x \in P \\ \zeta_r^i & \text{if } x \notin P \end{cases},$$

where $i \in \mathbb{Z}_r$ is the unique value such that $\zeta_r^i = x^{(|D_r/P|-1)/r} \pmod{P}$.

Remark 4.2.1 Recall that for $x \in \mathbb{Z}_p$, we have that x is a quadratic residue in \mathbb{Z}_p if and only if the Legendre symbol $\left(\frac{x}{p}\right) = 1$. In the same way, for $x \in D_r$, we have that x is an r th residue in D_r/P (i.e., $y^r = x \pmod{P}$ for some $y \in D_r$) if and only if $\left(\frac{x}{P}\right)_r = 1$.

Definition 4.2.3 For $A \subseteq D_r$, an ideal prime to r whose factorization into k prime ideals is $A = \prod_{i=1}^k P_i$, and for $x \in D_r$, we define the r th-power-residue symbol $\left(\frac{x}{A}\right)_r$ so that

$$\left(\frac{x}{A}\right)_r = \prod_{i=1}^k \left(\frac{x}{P_i}\right)_r.$$

If $N \in D_r$ and N is prime to r , we define $\left(\frac{x}{N}\right)_r = \left(\frac{x}{(N)}\right)_r$, where (N) is the ideal of D_r generated by N .

For $x, y \in D_r$ and ideals $I, J \subseteq D_r$ prime to r , the r th-power-residue symbol satisfies the following properties:

1. $\left(\frac{xy}{I}\right)_r = \left(\frac{x}{I}\right)_r \left(\frac{y}{I}\right)_r$ and $\left(\frac{x}{IJ}\right)_r = \left(\frac{x}{I}\right)_r \left(\frac{x}{J}\right)_r$.
2. If $x \in D_r$ is an r th residue in D_r/I , then $\left(\frac{x}{I}\right)_r = 1$; however, the converse need not be true. This is analogous to the fact that if x is a quadratic residue in \mathbb{Z}_N^\times , then the Jacobi symbol $\left(\frac{x}{N}\right) = 1$, though the converse need not be true.
3. Recall that $r \geq 3$ is odd. Then, $\left(\frac{x}{N}\right)_r = 1$ whenever x and N are both integers in \mathbb{Z} . In other words, the r th-power-residue symbol is degenerate when both $x, N \in \mathbb{Z}$. To see this, we use the fact that $\left(\frac{y}{I}\right)_r^\sigma = \left(\frac{y^\sigma}{I^\sigma}\right)_r$ for any Galois map $\sigma \in \text{Gal}(\mathbb{Q}[\zeta_r]/\mathbb{Q})$. Then, when $x, N \in \mathbb{Z}$, we see that $\left(\frac{x}{N}\right)_r$ is invariant under all Galois maps in $\text{Gal}(\mathbb{Q}[\zeta_r]/\mathbb{Q})$. But then, since $\left(\frac{x}{N}\right)_r$ is an r th root of unity and r is odd, we must have $\left(\frac{x}{N}\right)_r = 1$.

Next, we ask whether the r th-power-residue symbol is computable in polynomial time. One algorithm for doing so, due to Squirrel [54], is an r -dimensional generalization of the algorithm for computing the Jacobi symbols. The algorithm is based on the Eisenstein reciprocity law which can be viewed as a generalization of quadratic reciprocity (for the Jacobi symbol) to the r th-power-residue symbol. The following lemma (Algorithm 5.6 in [54]) is based on earlier work of Lenstra [37]:

Lemma 4.2.1 *For arbitrary relatively prime positive integers r and N and an arbitrary $x \in D_r$, $\left(\frac{x}{N}\right)_r$ can be computed in time polynomial in r and $\log N$.*

We present the idea behind an alternate algorithm next. In all the applications we have in mind the number r is either a small odd prime (*e.g.*, 3, 5, 7, ...) or is a product of distinct small odd primes. We note that when r is a product of distinct small primes, $r = r_1 r_2 \cdots r_k$, then to compute $\left(\frac{x}{I}\right)_r$ it suffices to compute $\left(\frac{x}{I}\right)_{r_1}, \dots, \left(\frac{x}{I}\right)_{r_k}$. However, we do not need this fact in the rest of the paper.

4.2.1 An Alternate Algorithm for Computing r th-Residue Symbols

We briefly sketch an alternate algorithm for computing the r th-residue symbol $\left(\frac{x}{I}\right)_r$ when I is a principal ideal of the form $I = (N)$ for some $N \in \mathbb{Z}$. The algorithm generalizes to arbitrary ideals J of D_r by an application of the Chinese remainder theorem.

Let r be an odd prime and let $N \in \mathbb{Z}$ be relatively prime to r . Let $x \in D_r$ and suppose that $x + \alpha$ generates a prime ideal in D_r for some $\alpha \in (N)$. Furthermore, suppose that $x + \alpha \in D_r$ is a primary element (*i.e.*, $x + \alpha$ is not a unit, $x + \alpha$ is relatively prime to r , and $x + \alpha$ is congruent to a rational integer modulo $(1 - \zeta_r)^2$). Then, using the Eisenstein reciprocity law [33, p. 207] and the definition of the r th-power-residue symbol, we get that

$$\left(\frac{x}{N}\right)_r = \left(\frac{x + \alpha}{N}\right)_r = \left(\frac{N}{x + \alpha}\right)_r \equiv N^{(\|(x+\alpha)\|-1)/r} \pmod{(x + \alpha)}.$$

This suggests the following algorithm, given $N \in \mathbb{Z}$ and $x \in D_r$ prime to N :

1. Pick random $\alpha \in (N)$ with norm less than some suitable bound B such that $x + \alpha$ is a primary element in D_r .
2. Test if $x + \alpha$ generates a prime ideal in D_r ; if not, go back to the previous step.
3. Find an $0 \leq i < r$ such that $\zeta_r^i \equiv N^{(\|(x+\alpha)\|-1)/r} \pmod{(x + \alpha)}$.

4. Output that $\left(\frac{x}{(N)}\right)_r = \zeta_r^i$.

The density of prime ideals in D_r tells us that the algorithm runs in time polynomial in r and $\log N + \log \|x\|$.

4.3 The Hardness of Distinguishing r th Residues

Next, we define r th roots of unity in \mathbb{Z}_N^\times and discuss the hardness of distinguishing r th residues from non-residues. Throughout this section, we let $N = pq$ where r divides both $p - 1$ and $q - 1$ but r^2 divides neither $p - 1$ or $q - 1$.

r th Roots of Unity. We denote by $Y_r(N)$ the r th roots of unity modulo N : $\{\mu \in \mathbb{Z}_N^\times : \mu^r = 1\}$.

Definition 4.3.1 *We say that an r th root of unity $\mu \in Y_r(N)$ is nondegenerate if for all prime factors t of N , the element μ viewed as an element of \mathbb{Z}_t^\times has order r . We denote by $Y_r^*(N) \subseteq Y_r(N)$ the nondegenerate r th roots of unity.*

Notice that when the factorization of r is known, testing for degeneracy is easy: $\mu \in \mathbb{Z}_N^\times$ is degenerate if and only if $\gcd(N, \mu^b - 1) > 1$ for some factor b of r where $1 \leq b < r$ (it suffices to test all b where r/b is a positive power of a prime). Throughout the paper, when referring to r th roots of unity in \mathbb{Z}_N^\times , we always refer to nondegenerate roots of unity.

In our case (*i.e.*, $N = pq$), the set of nondegenerate roots $Y_r^*(N)$ is of size $\phi(r)^2$. When r is prime, this set is of size $(r - 1)^2$.

Remark 4.3.1 *When $N = pq$, having two nondegenerate r th roots of unity (say, μ and λ) that are not a power of one another enables one to factor N (in time $O(r \log^2 N)$). To see this, observe that there must exist a k where $1 \leq k < r$ such that $\gcd(\mu^k \lambda - 1, N) = p$. By trying all $k \in \mathbb{Z}_r^\times$, we factor N in time $O(r \log^2 N)$.*

We define the r th-residuosity problem (RESIDUE_r) as follows:

Definition 4.3.2 For a fixed odd $r > 1$, an algorithm is said to solve the r th-residuosity problem (RESIDUE_r) if it can distinguish between the following pair of distributions, given N and r : (1) the uniform distribution on $PR_r(N)$, and (2) the uniform distribution on \mathbb{Z}_N^\times .

The RESIDUE_r problem is a generalization of the classic quadratic-residuosity problem modulo an RSA integer $N = pq$. The RESIDUE_r problem is believed to be hard for an RSA modulus $N = pq$ (when r divides $p-1$ and $q-1$) if no additional information about N is known. We will also need a slightly stronger hardness assumption which we call the (r, s) th residuosity assumption.

Definition 4.3.3 For a fixed odd $r, s > 1$ where r and s are relatively prime, an algorithm is said to solve the (r, s) th-residuosity problem ($\text{RESIDUE}_{r,s}$) if it can solve the r th-residuosity problem given both $N = pq$ (when r divides $p-1$ and $q-1$) and $\mu \in Y_s^*(N)$.

The $\text{RESIDUE}_{r,s}$ is believed to be hard if no additional information about N is known. In other words, RESIDUE_r remains hard even if a nondegenerate s th root of unity is known, for some s relatively prime to r .

Our first application for the r th-power-residue symbol shows that a nondegenerate r th root of unity, $\mu \in \mathbb{Z}_N^\times$, is sufficient for certain operations that previously were believed to require the factorization of N . For this to be interesting, we need that μ does not reveal the factorization of N . We state this more precisely below:

Assumption 4.3.1 Fix $r \in \mathbb{N}$. For $\lambda \in \mathbb{N}$, define $RSA_{\lambda,r}$ to be the set of RSA moduli that are a product of two λ -bit primes, $N = pq$, where r divides both $p-1$ and $q-1$. Then, for any probabilistic polynomial time algorithm \mathcal{A} and any polynomial $f: \mathbb{N} \rightarrow \mathbb{N}$ we have

$$\Pr[\mathcal{A}(N, r, \mu) \in \{p, q\}] < 1/f(\lambda) ,$$

where the probability is over the choice of $N \in RSA_{\lambda,r}$, $\mu \in Y_r^*(N)$, and the random bits of \mathcal{A} .

4.4 A Cryptosystem with a Reusable Modulus

Our first application of the r th-power-residue symbol is a cryptosystem with several interesting properties. We will work modulo an RSA modulus $N = pq$ and certain users will be able to decrypt messages without needing the factorization of N . We describe the cryptosystem by describing its **KeyGen**, **Encrypt**, and **Decrypt** algorithms.

KeyGen: Given a security parameter $\lambda \in \mathbb{N}$, the key-generation algorithm works as follows:

1. Let $r = r_1 r_2 \cdots r_k$ be the product of the first k odd primes. The cryptosystem will encrypt messages in \mathbb{Z}_r .
2. Let p and q be random λ -bit primes such that r divides both $p - 1$ and $q - 1$ but r^2 divides neither $p - 1$ or $q - 1$. Let $N = pq$.
3. Let $\mu \in \mathbb{Z}_N^\times$ be a nondegenerate r th root of unity, *i.e.*, $\mu \in Y_r^*(N)$.
4. Let I be the ideal of D_r generated by $I = (N, \zeta_r - \mu)$. Pick an arbitrary $g \in \mathbb{Z}_N^\times$ such that $(\frac{g}{I})_r$ has order r . A random $g \in \mathbb{Z}_N^\times$ will satisfy this with probability at least $\phi(r)/r$.
5. The public key is $\{N, g\}$. The factors $\{p, q\}$ are a strong trapdoor which enables decryption. We show that μ is a weak trapdoor which also enables decryption.

Encrypt: To encrypt a message $m \in \mathbb{Z}_r$ using the public key $\{N, g\}$, pick a random $h \in \mathbb{Z}_N^\times$ and compute $C = g^m h^r \in \mathbb{Z}_N^\times$.

Decrypt: Given a ciphertext $C = g^m h^r \in \mathbb{Z}_N^\times$, we decrypt C using the weak trapdoor μ by computing $m \bmod r_i$ for $i = 1, 2, \dots, k$. To compute $m \bmod r_i$, we do the following:

1. Let $\mu_i = \mu^{r/r_i}$ and let $I = (N, \zeta_{r_i} - \mu_i)$ be an ideal in D_{r_i} . Note that $\mu_i \in Y_{r_i}^*(N)$. Lemma 4.4.1 shows that I is an ideal of norm N and is a factor of the ideal (N) in D_{r_i} .

2. Using Lemma 4.2.1, find integers $0 \leq x_i, y_i < r_i$ such that

$$\left(\frac{C}{I}\right)_{r_i} = \zeta_{r_i}^{x_i} \quad \text{and} \quad \left(\frac{g}{I}\right)_{r_i} = \zeta_{r_i}^{y_i} .$$

Then, since $\left(\frac{C}{I}\right)_{r_i} = \left(\frac{g^m h^r}{I}\right)_{r_i} = \left(\frac{g}{I}\right)_{r_i}^m \left(\frac{h}{I}\right)_{r_i}^r = \left(\frac{g}{I}\right)_{r_i}^m$, and $\left(\frac{g}{I}\right)_{r_i}$ is of order r_i , it follows that $m = x_i y_i^{-1} \pmod{r_i}$. Thus, we can efficiently compute $m \pmod{r_i}$.

This completes the description of the system. We note that decryption using the strong trapdoor (the factorization of N) is the same as in Benaloh's system. We also note that by Assumption 4.3.1, the weak trapdoor μ does not reveal the factorization of N . This shows that the decrypter does not need the factorization of N to decrypt ciphertexts.

The purpose of this next lemma is to show that (N) factors into prime ideals. Though this uses basic techniques that can be found in texts such as [20, 33], we were unable to find a total presentation similar to this lemma.

Lemma 4.4.1 *If r is an odd prime and $\mu \in Y_r^*(N) \subseteq \mathbb{Z}_N^\times$, then*

$$(N) = \prod_{i=1}^{r-1} (N, \zeta_r - \mu^i) .$$

Proof. Throughout this proof, we use the following notation (for $i = 1, 2, \dots, r-1$): $I_i := (N, \zeta_r - \mu^i)$, $P_i := (p, \zeta_r - \mu_p^i)$, and $Q_i := (q, \zeta_r - \mu_q^i)$, where $\mu_p \in \mathbb{Z}_p^\times$ and $\mu_p = \mu \pmod{p}$ (similarly for q).

Theorem 4.8.13 in [20] shows that $(p) = \prod_{i=1}^{r-1} P_i$. Hence,

$$(N) = (p)(q) = \prod_{i=1}^{r-1} P_i Q_i ,$$

so, to complete the proof of the theorem, it is sufficient to prove that $I_i = P_i Q_i$ for all i .

We fix an i and show inclusion in each direction. Choose $a, b \in \mathbb{Z}_N$ such that $a = 0 \pmod{p}$, $a = p^{-1} \pmod{q}$, $b = q^{-1} \pmod{p}$, and $b = 0 \pmod{q}$. Set

$x := ap(\zeta_r - \mu_q^i) + bq(\zeta_r - \mu_p^i) \in P_iQ_i$ and notice that $x = \zeta_r - \mu^i \pmod{p}$ and $x = \zeta_r - \mu^i \pmod{q}$, so $x = \zeta_r - \mu^i \pmod{N}$.

If $y \in I_i$, we may write $y = \alpha N + \beta(\zeta_r - \mu^i)$ for some $\alpha, \beta \in D_r$. Hence, $y = \alpha pq + \beta x$, which is clearly in P_iQ_i .

Now we show the reverse inclusion. Set $c := (\zeta_r - \mu_p^i)(\zeta_r - \mu_q^i)$. First notice that, since $\mu_p^i + \mu_q^i = \mu^i + \mu_p^i\mu_q^i\mu^{-i}$ both modulo p and modulo q , they are equal modulo N . Hence,

$$c = \zeta_r^2 - \zeta_r(\mu_p^i + \mu_q^i) + \mu_p^i\mu_q^i = \zeta_r^2 - \zeta_r(\mu^i + \mu_p^i\mu_q^i\mu^{-i}) + \mu_p^i\mu_q^i = (\zeta_r - \mu_p^i\mu_q^i\mu^{-i})(\zeta_r - \mu^i)$$

modulo N . So $c = (\zeta_r - \mu_p^i\mu_q^i\mu^{-i})(\zeta_r - \mu^i) + \ell N$ for some $\ell \in \mathbb{Z}$; thus, c is clearly an element of $I_i = (N, \zeta_r - \mu^i)$.

Set $d := p(\zeta_r - \mu_q^i)$. Notice that, modulo p , we have that $d = 0 = p(\zeta_r - \mu^i)$. Modulo q , we have $d = p(\zeta_r - \mu^i)$, so we conclude that $d = p(\zeta_r - \mu^i)$ modulo N . Thus, $d \in I_i$. By a similar argument, we may conclude that $e := q(\zeta_r - \mu_p^i)$ is an element of I_i .

Let z be an arbitrary element of P_iQ_i and notice that we may choose $\alpha, \beta, \gamma, \delta \in D_r$ such that $z = \alpha pq + \beta p(\zeta_r - \mu_q^i) + \gamma q(\zeta_r - \mu_p^i) + \delta(\zeta_r - \mu_p^i)(\zeta_r - \mu_q^i)$. Since $z = \alpha N + \beta d + \gamma e + \delta c$, we conclude that $z \in I_i$. \square

Security. Semantic security follows from the hardness of the r th-residuosity problem (Definition 4.3.2) in a standard way. Suppose that algorithm \mathcal{A} is able to break the semantic security of the system. Then, algorithm \mathcal{A} , given the public key $\{N, g\}$, will output two messages $m_0, m_1 \in \mathbb{Z}_r$. The algorithm is given an encryption of m_b for a random $b \in \{0, 1\}$ and will be able to correctly output b with probability at least $\frac{1}{2} + \varepsilon$, for some $\varepsilon > 0$.

Lemma 4.4.2 *Using Algorithm \mathcal{A} defined above, it is possible to solve the r th-residuosity problem with advantage at least ε in time polynomial in the running time of \mathcal{A} .*

Proof. The lemma is a special case of a more general lemma (Lemma 4.4.3) which we prove in the next section. Lemma 4.4.2 follows by setting $n = 1, s_1 = 1$ in

Lemma 4.4.3. □

We note that the system is additively homomorphic. Given two ciphertexts $C_1, C_2 \in \mathbb{Z}_N^\times$ that are encryptions of $m_1, m_2 \in \mathbb{Z}_r$ respectively, we see that $C_1 C_2 \in \mathbb{Z}_N^\times$ is an encryption of $m_1 + m_2$. This property is useful for anonymous electronic voting. The weak-trapdoor property means that the tabulation center need only be given the weak trapdoor μ . It does not need to be given the factorization of N .

4.4.1 Partial Decryption

Since the weak decryption trapdoor does not reveal the factorization of N , we can control which segments of the message the recipient can decrypt. Consider a public-key system where **KeyGen** and **Encrypt** are the same as in the previous section. The public key is $N = pq$ and g . Recall that $r = r_1 r_2 \cdots r_k$ is the product of the k first odd primes and r divides both $p - 1$ and $q - 1$.

Let $s \in \mathbb{N}$ be a factor of r . Suppose that we give the decrypter the key $\mu_s = \mu^{r/s}$, where $\mu \in Y_r^*(N)$ is the r th root of unity generated by algorithm **KeyGen**. Then, given a ciphertext $C = g^m h^r \in \mathbb{Z}_N$, the decrypter can use **Decrypt** of the previous section to recover $m \bmod r_i$ for any r_i that divides s . Hence, the decrypter can recover $m \bmod s$. We claim that the rest of the message, namely $m \bmod r/s$, remains semantically secure from this decrypter. This is stated in the following lemma:

Lemma 4.4.3 *Let $N = pq, g$ be a public key generated by the **KeyGen** algorithm. Let $\mu \in Y_r^*(N)$. Suppose that there are n decrypters, and decrypter i has trapdoor $\mu_{s_i} = \mu^{r/s_i}$ for some s_i dividing r . Let $s = \text{lcm}(s_1, s_2, \dots, s_n)$. Then, even if all n decrypters collude, the system remains semantically secure for $m \bmod r/s$, assuming the $(r/s, s)$ -th-residuosity problem is hard.*

Proof. Suppose that an algorithm \mathcal{A} is able to break the semantic security of the system. Then algorithm \mathcal{A} would work as follows. It is given the public key $\{N, g\}$ and the private keys $\mu_{s_1}, \mu_{s_2}, \dots, \mu_{s_n}$, where $\mu_{s_i} = \mu^{r/s_i}$. Let $s = \text{lcm}(s_1, s_2, \dots, s_n)$. The algorithm outputs two messages $m_0, m_1 \in \mathbb{Z}_r$ such that $m_0 = m_1 \bmod s$ and is then given an encryption of m_b under $\{N, g\}$ for a random $b \in \{0, 1\}$. The algorithm

outputs $b' \in \{0, 1\}$ and we have that $b = b'$ with probability at least $\frac{1}{2} + \varepsilon$, for some $\varepsilon > 0$.

We now show how to use \mathcal{A} to solve the $(r/s, s)$ th-residuosity problem. We are given $N = pq$, $\mu_s \in Y_s^*(N)$, and an $x \in \mathbb{Z}_N^\times$ that is either uniform in \mathbb{Z}_N^\times or uniform in $\text{PR}_{r/s}(N)$. To decide which distribution x is from, do the following:

1. Pick a random $g \in \mathbb{Z}_N^\times$ and give \mathcal{A} the public key $\{N, g\}$ as well as the n private keys $\mu_s^{s/s_1}, \mu_s^{s/s_2}, \dots, \mu_s^{s/s_n}$.
2. Algorithm \mathcal{A} outputs two messages $m_0, m_1 \in \mathbb{Z}_r$ such that $m_0 = m_1 \pmod s$. Pick a random $b \in \{0, 1\}$ and compute $C = g^{m_b} x^s \in \mathbb{Z}_N$. Give \mathcal{A} the challenge ciphertext C .
3. Algorithm \mathcal{A} will respond with $b' \in \{0, 1\}$. If $b = b'$, say that x is uniform in $\text{PR}_{r/s}(N)$. Otherwise, say that x is uniform in \mathbb{Z}_N^\times .

Suppose that the g picked in Step 1 is valid for a public key (as described in the **KeyGen** algorithm). Then, (1) if x is uniform in \mathbb{Z}_N^\times , then x^s is a uniform s th residue in \mathbb{Z}_N^\times , and (2) if x is uniform in $\text{PR}_{r/s}(N)$, then C is a proper encryption of m_b . By definition of \mathcal{A} , in case (2), $b = b'$ with probability at least $\frac{1}{2} + \varepsilon$. In case (1), the challenge ciphertext C is independent of b and therefore $b = b'$ with probability $\frac{1}{2}$. Therefore, when g is valid for a public key, we have advantage at least ε in distinguishing (r/s) th residues from non-residues. Since the probability that g is valid for a public key is at least $\phi(r)/r$, we see that repeating this experiment $\Omega(r/\phi(r))$ times with independently chosen g and taking a majority vote will produce the correct result with probability at least ε . \square

The lemma shows that we can give different decrypters different weak trapdoors that enable them to recover different parts of the plaintext while the remaining parts will remain semantically secure. The ciphertext is a single element in \mathbb{Z}_N .

Applications. Some applications of this mechanism include:

- **Partial escrow decryption:** An escrow agent can be given a weak trapdoor that enables decryption of part of the ciphertext relevant to the agent (*e.g.*, the recipient's name), but the rest of the message remains hidden from the escrow agent.

Utilizing this in the context of a cryptographic counter, users can increment or decrement the counter using the public key and the additively homomorphic property of the system, while the value of the counter remains secret. The escrow agent can be given a weak trapdoor that enables him to monitor a few bits of the counter (*i.e.*, the value of the counter modulo a predefined number) without ever learning the full value of the counter.

- Weak broadcast encryption: A single element in \mathbb{Z}_N^\times can be used to implement *public-key broadcast encryption* to a small number of recipients (*e.g.*, 2 or 3). For example, let $r = r_1 r_2 r_3$ with $r_1 < r_2 < r_3$. For $i = 1, 2, 3$, recipient i is given a nondegenerate r_i th root of unity in \mathbb{Z}_N^\times . Then, we can broadcast $m' \in [1, r_1]$ to any subset of recipients by creating a message $m \in \mathbb{Z}_r$ such that $m = m' \pmod{r_i}$ for users i that are valid recipients, but $m = 0 \pmod{r_i}$ for users i that are not. The ciphertext is a single element of \mathbb{Z}_N^\times , namely $C = g^m h^r \pmod{N}$ (for a randomly chosen $h \in \mathbb{Z}_N^\times$).

4.4.2 Encrypted and Signed Messages

By giving the weak trapdoor (μ) to one user and the strong trapdoor (the factorization of N) to another user, we can achieve a form of signcryption [64, 3] for short messages. We describe this briefly.

Let r be the product of the first k primes greater than 3. Let $N = pq$ be such that r divides both $p - 1$ and $q - 1$, but 3 does not divide either. Let μ be a nondegenerate r th root of unity in \mathbb{Z}_N^\times . Suppose Alice has the factorization of N while Bob has μ . Then Alice can generate RSA signatures using N as a public key (she can compute cube roots in \mathbb{Z}_N). Similarly, Bob can use μ to decrypt messages created with the system of the previous section, but he cannot generate RSA signatures modulo N . To send an encrypted and signed message m to Bob, Alice computes

$$C = (g^{\text{PAD}(m)} h^r)^{1/3} \pmod{N},$$

where $\text{PAD}: \{0, 1\}^n \rightarrow \mathbb{Z}_r$ is a universal padding function designed to provide chosen-ciphertext security and existential unforgeability in the random oracle model (as in [23]). We do not explore the necessary padding function here. Bob can recover m by decrypting the ciphertext C^3 . At the same time Bob obtains Alice's non-repudiable signature on m — the value C functions as the signature. Note that to convince a third party of the validity of this signature on m , Bob must reveal his private key μ to the third party.

4.5 Factoring Integers of the Form $N = pq^r$

We present another application of the power-residue symbol, this time *not* requiring an r th root of unity (μ).

Peralta and Okamoto [44] proposed an improvement to Lenstra's elliptic-curve method [36] for factoring numbers of the form $N = pq^2$. Peralta and Okamoto's result has the same asymptotic complexity as results which use the fast Fourier transform (FFT) (*e.g.*, [40]), though their algorithm is simpler and seems faster in practice. We show how to apply these ideas to numbers of the form $N = pq^r$ for $r > 2$. We assume that the reader has some familiarity with the elliptic-curve factoring method (ECM).

Definition 4.5.1 ([6]) *A positive integer N is said to be semi-smooth with respect to B_1 and B_2 if all the prime factors of N are bounded above by $B_1 \leq B_2$, with the possible exception of one factor which is bounded above by B_2 and has multiplicity 1.*

Next, we define the r th-power-residue signature of an element $x \in \mathbb{Z}_N^\times$:

Definition 4.5.2 *For relatively prime positive integers r and N , a positive integer k , and $x \in \mathbb{Z}_N^\times$, define the r th-power-residue signature (of length k) of x to be (recall that $\zeta_r = e^{2\pi i/r} \in \mathbb{C}$):*

$$\mathbf{J}_r^{(k)}(x, N) = \left(\left(\frac{x + \zeta_r}{N} \right)_r, \left(\frac{x + 2\zeta_r}{N} \right)_r, \dots, \left(\frac{x + k\zeta_r}{N} \right)_r \right).$$

In calculating the running time of our algorithm (**Factor**), we assume that we have an elliptic curve \mathcal{C} whose order over \mathbb{Z}_N^\times is semi-smooth. In addition to N and \mathcal{C} , the

algorithm takes as input two further parameters, B (which, without loss of generality, we assume to be prime) and $m \in \mathbb{N}$. (Additionally, we require $\gcd(N, r) = 1$, so the r th-power-residue symbols used will be defined.)

Factor (N, \mathcal{C}, B, m) :

1. Let P be a random point on the curve \mathcal{C} (in practice, one first chooses a random P and then chooses a random curve \mathcal{C} with P on it).
2. Set $Q := L \cdot P \in \mathcal{C}$, where $L = 2^{\lceil \log_2 N \rceil} 3^{\lceil \log_3 N \rceil} 5^{\lceil \log_5 N \rceil} \dots B^{\lceil \log_B N \rceil}$ (which never needs to be explicitly computed).
3. Randomly choose $t_1, t_2, \dots, t_m \in \{1, 2, \dots, B\}$.
4. Set $R_i = (x_i, y_i) := t_i \cdot Q \in \mathcal{C}$ for $i = 1, 2, \dots, m$.
5. Compute $s_i := \mathbf{J}_r^{(\lceil \log_r m \rceil)}(x_i, N)$.
6. When $s_i = s_j$, we compute $\gcd(x_i - x_j, N)$ and output any nontrivial value.

Before proving that **Factor** finds a factor of N , we state a natural number theoretic conjecture upon which the theorem relies. The conjecture states that no two elements $x \neq w \in \mathbb{F}_p$ have the same r th residue signature.

Conjecture 4.5.1 *If p is prime and $p < B$ for some integer B , then*

$$\mathbf{J}_r^{\left(\lceil \frac{1}{2} \log_2 B \rceil\right)}(x, p) = \mathbf{J}_r^{\left(\lceil \frac{1}{2} \log_2 B \rceil\right)}(w, p) \quad \text{if and only if } x \equiv w \pmod{p} .$$

We note that Conjecture 4.5.1 is analogous to the conjecture for Jacobi symbols upon which [44] relies.

Theorem 4.5.2 *Given an integer $N = pq^r$ and a curve \mathcal{C} whose order is semi-smooth with respect to B_1 and B_2 , **Factor** $(N, \mathcal{C}, B_1, \lceil \sqrt{B_2} \rceil)$ outputs, with probability at least $1/4$, a nontrivial factor of N .*

Proof. By viewing the (x_i, y_i) as independently chosen random elements of \mathcal{C} , the birthday paradox tells us that with probability about $1/2$ (actually, the choice of $m = \lceil \sqrt{B_2} \rceil$ gives, asymptotically, a probability of $1 - 1/\sqrt{e}$) the algorithm will find $i \neq j$ such that $s_i = s_j$.

Notice that, for any $x \in D_r$, $\left(\frac{x}{N}\right)_r = \left(\frac{x}{p}\right)_r \left(\frac{x}{q}\right)_r^r = \left(\frac{x}{p}\right)_r$ (unless $q|x$, in which case we can compute $\gcd(x, N) = q$ and factor N); thus, we know that, for $i \in \{1, 2, \dots, m\}$, $\mathbf{J}_r^{\lceil \frac{1}{2} \log_r B_2 \rceil}(x_i, N) = \mathbf{J}_r^{\lceil \frac{1}{2} \log_r B_2 \rceil}(x_i, p)$. Thus, when $s_i = s_j$, we know that $\mathbf{J}_r^{\lceil \frac{1}{2} \log_r B_2 \rceil}(x_i, p) = \mathbf{J}_r^{\lceil \frac{1}{2} \log_r B_2 \rceil}(x_j, p)$ and — assuming Conjecture 4.5.1 — that $x_i = x_j \pmod{p}$. By viewing each of the (x_i, y_i) as independently chosen random elements of \mathcal{C} , we conclude that whenever $x_i = x_j \pmod{p}$, the probability that $x_i \neq x_j \pmod{N}$ (and, hence, a nontrivial factor of N will be output by **Factor**) is $1 - \frac{1}{|\mathbb{Z}_{q^r}^\times|} = 1 - \frac{1}{q^{r-1}(q-1)}$.

Hence, the algorithm outputs a nontrivial factor of N with probability approximately $\left(1 - \frac{1}{q^{r-1}(q-1)}\right) \left(1 - \frac{1}{\sqrt{e}}\right) > \frac{1}{4}$. \square

Analysis of the run time of **Factor** parallels the analysis of Peralta-Okamoto: **Factor** will find p in time $L_p \left[\frac{1}{2}, \sqrt{2}\right] = O\left(e^{\sqrt{2} \log p \log \log p}\right)$ (with probability at least $\frac{1}{4}$), as with Peralta and Okamoto's Algorithm 7 (and all known variants of the ECM).

4.6 Open Problems

Several natural extensions to the cryptosystem described in Section 4.4 present interesting open problems; *e.g.*:

- The Pallier system can be viewed as a more efficient variant of Benaloh's system. It is interesting to see whether the techniques of this paper can produce a weak trapdoor for that system as well. In other words, can one decrypt ciphertexts in the Pallier system using a secret key that does not reveal the factorization of N ?
- We restricted ourselves to odd, square-free r ; if $r = 2^d$, can we still decrypt messages with the help of a 2^d th root of unity? Can the 2^d th-power-residue symbol be computed in time polynomial in d and $\log N$?

4.7 Conclusions

We studied a number of applications of the power-residue symbol to cryptography. First, we showed that a slight variant of the Benaloh system has a weak trapdoor. That is, decryption can be done using a secret key that does not reveal the factorization of N . We discussed a few implications of this fact including partial decryption of ciphertexts (not possible in the original Benaloh system) and weak signcryption. We also presented another application of the power-residue symbol: we showed that it can be used to improve the ECM factoring algorithm for integers of the form $N = pq^r$ (when $r > 2$). The power-residue symbol and Eisenstein reciprocity are powerful tools. We hope that they will find other applications in cryptography.

Bibliography

- [1] K.S. Abdukhalikov and C. Kim, “On the Security of the Hashing Scheme Based on SL_2 .” *Fast Software Encryption: FSE '98 (LNCS 1372)*, pp. 93–102, 1998.
- [2] N. Alon and Y. Roichman, “Random Cayley Graphs and Expanders.” *Random Structures and Algorithms* **5**, pp. 271–284, 1994.
- [3] J.H. An, Y. Dodis, and T. Rabin, “On the Security of Joint Signature and Encryption.” *Advances in Cryptology: EUROCRYPT 2002 (LNCS 2332)*, pp. 83–107, 2002.
- [4] L. Babai, “Local Expansion of Vertex Transitive Graphs and Random Generation of Finite Groups.” *Symposium on the Theory of Computing (STOC '91)*, pp. 164–174, 1991.
- [5] E. Bach, “Toward a Theory of Pollard’s Rho Method.” *Information and Computation* **90**(2), pp. 139–155, 1991.
- [6] E. Bach and R. Peralta, “Asymptotic Semismoothness Properties.” *Mathematics of Computation* **65**, pp. 1701–1715, 1996.
- [7] M. Bellare, O. Goldreich, and S. Goldwasser, “Incremental Hashing: The Case of Hashing and Signing.” *Advances in Cryptology: CRYPTO '94 (LNCS 839)*, pp. 216–233, 1994.
- [8] M. Bellare and D. Micciancio, “A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost.” *Advances in Cryptology: EUROCRYPT '97 (LNCS 1233)*, pp. 163–192, 1997.

- [9] J. Benaloh, “Verifiable Secret-Ballot Elections.” Ph.D. thesis, Yale University, 1987.
- [10] J. Benaloh and D. Tuinstra, “Receipt-Free Secret-Ballot Elections.” *Symposium on the Theory of Computing (STOC '94)*, pp. 544–553, 1994.
- [11] J. Benaloh and M. Yung, “Distributing the Power of a Government to Enhance the Privacy of Voters.” *Symposium on Principles of Distributed Computing (PODC '86)*, pp. 52–62, 1986.
- [12] B. Bollobas, *Modern Graph Theory*. Graduate Texts in Mathematics **184**, Springer-Verlag, Berlin, 1998.
- [13] D. Boneh, G. Durfee, and N. Howgrave-Graham, “Factoring $N = p^r q$ for Large r .” *Advances in Cryptology: CRYPTO '99 (LNCS 1666)*, pp. 326–337, 1999.
- [14] D. Boneh and M. Franklin, “Identity Based Encryption from the Weil Pairing.” *Advances in Cryptology: CRYPTO 2001 (LNCS 2319)*, pp. 213–229, 2001.
- [15] D. Boneh and M. Franklin, “Identity Based Encryption from the Weil Pairing.” *Cryptology ePrint Archive*, Report 2001/090, 2001. <http://eprint.iacr.org/2001/090/>
- [16] D. Boneh, B. Lynn, and H. Shacham, “Short Signatures from the Weil Pairing.” *Advances in Cryptology: ASIACRYPT 2001 (LNCS 2248)*, pp. 514–532, 2001.
- [17] A. Broder and E. Shamir, “On the Second Eigenvalue of Random Regular Graphs.” *Symposium on the Foundations of Computer Science (FOCS '87)*, pp. 286–294, 1987.
- [18] C. Charney and J. Pieprzyk, “Attacking the SL_2 Hashing Scheme.” *Advances in Cryptology: ASIACRYPT '94 (LNCS 917)*, pp. 322–330, 1994.
- [19] C. Cocks, “An Identity Based Encryption Based on Quadratic Residues.” *Cryptography and Coding (LNCS 2260)*, pp. 360–363, 2002.

- [20] H. Cohen, *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics **138**, Springer-Verlag, Berlin, 1993.
- [21] J. Cohen and M. Fischer, “A Robust and Verifiable Cryptographically Secure Election Scheme.” *Symposium on Foundations of Computer Science (FOCS '85)*, pp. 372–382, 1985.
- [22] J. Coron, “On the Exact Security of Full Domain Hash.” *Advances in Cryptology: CRYPTO 2000 (LNCS 1880)*, pp. 229–235, 2000.
- [23] J-S. Coron, M. Joye, D. Naccache, P. Paillier, “Universal Padding Schemes for RSA.” *Advances in Cryptology: CRYPTO 2002 (LNCS 2442)*, pp. 226–241, 2002.
- [24] I. Damgård, “Collision-Free Hash Functions, Public Key Signature Schemes.” *Advances in Cryptology: EUROCRYPT '87 (LNCS 304)*, pp. 203–216, 1987.
- [25] C. Dou and M. Hildebrand, “Enumeration and Random Random Walks on Finite Groups.” *Annals of Probability* **24**(2), pp. 987–1000, 1996.
- [26] T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.” *Advances in Cryptology: CRYPTO '84 (LNCS 196)*, pp. 10–18, 1984.
- [27] J.A. Fill, “Eigenvalue Bounds on Convergence to Stationary for Nonreversible Markov Chains, with an Application to the Exclusion Process.” *Annals of Applied Probability* **1**(1), pp. 62–87, 1991.
- [28] E. Fujisaki and T. Okamoto, “Secure Integration of Asymmetric and Symmetric Encryption Schemes.” *Advances in Cryptology: CRYPTO '99 (LNCS 1666)*, pp. 537–554, 1999.
- [29] C. Gentry and A. Silverberg, “Hierarchical ID-Based Cryptography.” *Advances in Cryptology: ASIACRYPT 2002 (LNCS 2501)*, pp. 548–566, 2002.
- [30] C. Gentry and A. Silverberg, “Hierarchical ID-Based Cryptography”, *Cryptology ePrint Archive*, Report 2002/056, 2002. <http://eprint.iacr.org/2002/056/>

- [31] J. Horwitz and B. Lynn, “Toward Hierarchical Identity-Based Encryption.” *Advances in Cryptology: EUROCRYPT 2002 (LNCS 2332)*, pp. 466–481, 2002.
- [32] J. Horwitz and R. Venkatesan, “Random Cayley Digraphs and the Discrete Logarithm.” *Algorithmic Number Theory Symposium V (ANTS-V)*, Lecture Notes in Computer Science vol. 2369, pp. 100–114, 2002.
- [33] K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*. Graduate Texts in Mathematics **84**, 2nd ed., Springer-Verlag, Berlin, 1990.
- [34] A. Joux, “A One Round Protocol for Tripartite Diffie-Hellman.” *Algorithmic Number Theory : 4th International Symposium, ANTS-IV (LNCS 1838)*, pp. 385–394, 2000.
- [35] M. Kasahar, K. Ohgishi, and R. Sakai, “Cryptosystems Based on Pairing.” *The 2001 Symposium on Cryptography and Information Security*, Oiso, Japan, 2001.
- [36] H.W. Lenstra, “Factoring Integers with Elliptic Curves.” *Annals of Mathematics* **126**, pp. 649–673, 1987.
- [37] H.W. Lenstra, “Computing Jacobi Symbols in Algebraic Number Fields.” *Nieuw Archief voor Wiskunde* **13**(3), pp. 421–426, 1995.
- [38] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, New York, 1977.
- [39] M. Mihail, “Conductance and Convergence of Markov Chains — a Combinatorial Treatment of Expanders.” *Symposium on the Foundations of Computer Science (FOCS '89)*, pp. 526–531, 1989.
- [40] P.L. Montgomery and R.D. Silverman, “An FFT Extension to the $p-1$ Factoring Algorithm.” *Mathematics of Computation* **54**, pp. 839–854, 1990.
- [41] D. Naccache and J. Stern, “A New Cryptosystem Based on Higher Residues.” *ACM Communications and Computer Security (CCS '98)*, pp. 59–66, 1998.

- [42] T. Okamoto, S. Uchiyama, “A New Public-Key Cryptosystem as Secure as Factoring.” *Advances in Cryptology: EUROCRYPT '98 (LNCS 1403)*, pp. 308–318, 1998.
- [43] P. Pallier, “Public-Key Cryptosystems Based on Composite Degree Residue Classes.” *Advances in Cryptology: EUROCRYPT '99 (LNCS 1592)*, pp. 223–238, 1999.
- [44] R. Peralta and E. Okamoto, “Faster Factoring of Integers of a Special Form.” *TIEICE: IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences* **E79-A**(4), pp. 489–493, 1996.
- [45] S.C. Pohlig and M.E. Hellman, “An Improved Algorithm for Computing Logarithms over $GF(p)$ and Its Cryptographic Significance.” *IEEE Transactions on Information Theory* **24**, pp. 106–110, 1978.
- [46] J.M. Pollard, “Monte Carlo Methods for Index Computation (mod p).” *Mathematics of Computation* **32**(143), pp. 918–924, 1978.
- [47] J.M. Pollard, “Kangaroos, Monopoly, and Discrete Logarithms.” *Journal of Cryptology* **13**, pp. 437–447, 2000.
- [48] M.O. Rabin, “Digitalized Signatures and Public-Key Functions as Intractible as Factorization”, MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.
- [49] Y. Roichman, “On Random Random Walks.” *Annals of Probability* **24**(2), pp. 1001–1011, 1996.
- [50] E. Seneta, *Non-negative Matrices and Markov chains*, 2nd ed. Springer Series in Statistics, Springer-Verlag, 1981.
- [51] A. Shamir, “Identity-Based Cryptosystems and Signature Schemes.” *Advances in Cryptology: CRYPTO '84 (LNCS 196)*, pp. 47–53, 1985.

- [52] V. Shoup, “Lower Bounds for Discrete Logarithms and Related Problems.” *Advances in Cryptology: EUROCRYPT '97 (LNCS 1233)*, pp. 256–266, 1997.
- [53] V. Shoup, *NTL: A Library for Doing Number Theory*.
<http://www.shoup.net/ntl/>
- [54] D. Squirrel, undergraduate thesis, Reed College, 1997.
- [55] T. Takagi, “Fast RSA-type Cryptosystem Modulo p^kq .” *Advances in Cryptology: CRYPTO '98 (LNCS 1462)*, pp. 318–326, 1998.
- [56] E. Teske, “Speeding Up Pollard’s Rho Method for Computing Discrete Logarithms.” *Algorithmic Number Theory Symposium III (LNCS 1423)*, pp. 541–554, 1998.
- [57] E. Teske, “On Random Walks for Pollard’s Rho Method.” *Mathematics of Computation* **70**, pp. 809–825, 2001.
- [58] J.-P. Tillich and G. Zémor, “Hashing with SL_2 .” *Advances in Cryptology: CRYPTO '94 (LNCS 839)*, pp. 40–49, 1994.
- [59] J.-P. Tillich and G. Zémor, “Group-Theoretic Hash Functions.” *First French-Israeli Workshop on Algebraic Coding*, pp. 90–110, 1994.
- [60] P.C. van Oorschot and M.J. Weiner, “Parallel Collision Search with Cryptanalytic Applications.” *Journal of Cryptology* **12**, pp. 1–28, 1999.
- [61] V. Vazirani, “Rapidly Mixing Markov Chains.” In B. Bollobas, editor, *Probabilistic Combinatorics and Its Applications* **44**, pp. 99–121. AMS, Providence, Rhode Island, 1991.
- [62] E. Verheul, “Evidence That XTR Is More Secure than Supersingular elliptic curve cryptosystems.” *Advances in Cryptology: EUROCRYPT 2001 (LNCS 2045)*, pp. 195–210, 2001.
- [63] E. Verheul, “Self-Blindable Credential Certificates from the Weil Pairing.” *Advances in Cryptology: ASIACRYPT 2001 (LNCS 2248)*, pp. 533–551, 2001.

- [64] Y. Zheng, “Digital Signcryption or How to Achieve $\text{Cost}(\text{Signature} \ \& \ \text{Encryption}) \ll \text{Cost}(\text{Signature}) + \text{Cost}(\text{Encryption})$.” *Advances in Cryptology: CRYPTO '97 (LNCS 1294)*, pp. 165–179, 1997.
- [65] ISO/IEC 9594-8, “Information Technology — Open Systems Interconnection — The Directory: Authentication Framework”, International Organization for Standardization, Geneva, Switzerland, 1995 (equivalent to ITU-T Recommendation X.509, 1993).