The Theory and Applications of Homomorphic Cryptography

by

Kevin Henry

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

Abstract

Homomorphic cryptography provides a third party with the ability to perform simple computations on encrypted data without revealing any information about the data itself. Typically, a third party can calculate one of the encrypted sum or the encrypted product of two encrypted messages. This is possible due to the fact that the encryption function is a group homomorphism, and thus preserves group operations. This makes homomorphic cryptosystems useful in a wide variety of privacy preserving protocols.

A comprehensive survey of known homomorphic cryptosystems is provided, including formal definitions, security assumptions, and outlines of security proofs for each cryptosystem presented. Threshold variants of several homomorphic cryptosystems are also considered, with the first construction of a threshold Boneh-Goh-Nissim cryptosystem given, along with a complete proof of security under the threshold semantic security game of Fouque, Poupard, and Stern. This approach is based on Shoup's approach to threshold RSA signatures, which has been previously applied to the Paillier and Damgård-Jurik cryptosystems. The question of whether or not this approach is suitable for other homomorphic cryptosystems is investigated, with results suggesting that a different approach is required when decryption requires a reduction modulo a secret value.

The wide variety of protocols utilizing homomorphic cryptography makes it difficult to provide a comprehensive survey, and while an overview of applications is given, it is limited in scope and intended to provide an introduction to the various ways in which homomorphic cryptography is used beyond simple addition or multiplication of encrypted messages. In the case of strong conditional oblivious transfer, a new protocol implementing the greater than predicate is presented, utilizing some special properties of the Boneh-Goh-Nissim cryptosystem to achieve security against a malicious receiver.

Acknowledgements

I would like to thank my supervisor, Doug Stinson, and the other members of the CrySP lab for their help and support during my studies. I would also like to thank my readers, Alfred Menezes and David Jao, for their helpful comments and suggestions.

Contents

List of Tables

Chapter 1

Introduction

Homomorphic cryptosystems are cryptosystems whose encryption function is a homomorphism, and thus preserves group operations performed on ciphertexts. Depending on the properties of the cryptosystem, this usually allows a third party to take two ciphertexts $e_K(m_1)$ and $e_K(m_2)$, and, without knowledge of any secret information, calculate one of $e_K(m_1 + m_2)$ or $e_K(m_1m_2)$. The ability to perform simple computations on ciphertexts allows for a variety of simple privacy preserving protocols to be built upon homomorphic cryptosystems. Unfortunately, the additional structure provided by a homomorphic cryptosystem also places limits on their security.

Homomorphic cryptosystems are a special instance of general privacy homomorphisms, a concept introduced by Rivest, Adleman, and Dertouzous in 1978 [86], originally envisioned as a method to allow expensive computations to be performed by a potentially untrusted third party. If a user could take a problem defined in one algebraic system and encode it into a problem in a different algebraic system such that decoding back to the original algebraic system is hard, then the user could encode expensive computations and send them to the untrusted party. This untrusted party then performs the corresponding computation in the second algebraic system, returning the result to the user. Upon receiving the result, the user can decode it into a solution in the original algebraic system, while the untrusted party learns nothing of what computation was actually performed.

In many applications, such as cryptographic voting protocols, it is desirable to distribute trust among the participants involved such that some pre-defined threshold, potentially all participants, must agree to cooperate in order to decrypt a message. In the context of a voting protocol, the threshold might be chosen to ensure that at least one official from each political party involved must agree to cooperate in decryption. Many homomorphic cryptosystems have efficient threshold variants, further bolstering their use in privacy preserving protocols.

This thesis provides a survey of security notions for homomorphic cryptosystems in an abstract setting, as well as a comprehensive survey of known homomorphic cryptosystems, including outlines of proofs of security, known variants, and a summary of the computational problems relied on by each cryptosystem. The survey is extended to threshold variants of these cryptosystems, and the open problem of creating threshold variants for several homomorphic cryptosystems is investigated with both positive and negative results. To conclude, a brief overview of how homomorphic cryptography is applied within privacy preserving protocols is provided.

1.1 Contributions

The survey of homomorphic cryptography given in Chapter 3 contains a summary of all known homomorphic cryptosystems, as well as the many variants of these cryptosystems. To remain self-contained, outlines of security proofs are provided for each cryptosystem and a description of the computational problem each cryptosystem relies on is also given. This survey contains several variants and a major cryptosystem, The Boneh-Goh-Nissim cryptosystem, not included in previous surveys, such as [72].

In regards to threshold homomorphic cryptosystems, the first secure construction of a threshold variant of the Boneh-Goh-Nissim cryptosystem is demonstrated, including a full security proof. This threshold variant is based on an approach by Shoup [95], which has previously been used to construct other threshold homomorphic cryptosystems, and is shown to be secure with respect to the threshold semantic security game given by Fouque, Poupard, and Stern [48]. An open problem posed by Fouque, Poupard, and Stern, the creation of threshold variants for many other homomorphic cryptosystems, is investigated with respect to Shoup's approach, and it is shown that secure threshold variants utilizing Shoup's approach in conjunction with a secondary (required) protocol do not exist. The general problem of creating secure threshold variants of these cryptosystems remains open.

When presenting some applications of homomorphic cryptography, a new strong conditional oblivious transfer protocol for the greater than predicate is provided that is secure against a malicious receiver. This new protocol replaces the Paillier cryptosystem with the Boneh-Goh-Nissim cryptosystem, and exploits the fact that a single homomorphic multiplication on BGN ciphertexts is possible. This allows the sender to "sanitize" any maliciously crafted messages sent by the receiver, using the same approach Boneh, Goh, and Nissim used when evaluating 2-DNF formulas [14].

1.2 Organization

The remainder of the introduction presents a basic overview of cryptography, formally defining a cryptosystem and the various notions of security. An overview of homomorphic cryptography in an abstract setting is then given, beginning with an introduction to privacy homomorphisms. Once the basic definitions are established, the theoretical limitations of security of homomorphic cryptosystems is presented, followed by known attacks against homomorphic cryptosystems in a general setting, and new alternate definitions of security for homomorphic cryptosystems.

Chapter 3 presents an overview of the many computational problems that the security of homomorphic cryptosystems rely on, citing the best known approaches to solving each problem. This is followed by a comprehensive survey of known homomorphic cryptosystems, in order of discovery, with each major cryptosystem formally stated alongside an outline of the security proof, a discussion of any variants of the cryptosystem, and a summary of the homomorphic properties of the cryptosystem.

Known threshold variants of homomorphic cryptosystems are omitted from the survey in Chapter 3 and are instead summarized in Chapter 4. Shoup's threshold RSA signature scheme is presented, followed by the threshold homomorphic cryptosystems that have built off his approach. A new construction of a threshold BGN cryptosystem is presented, followed by a discussion on the impossibility of directly applying Shoup's approach to homomorphic cryptosystems that currently have no known threshold variant.

Before concluding, a brief overview of some protocols utilizing homomorphic cryptosystems is given in Chapter 5. This includes a new construction of a strong conditional oblivious transfer protocol implementing the greater than predicate that is secure against a malicious receiver. A sample implementation of two privacy preserving protocols is also provided in Appendix A.

1.3 Introduction to Cryptography

In general, a cryptosystem provides a method for transforming one message, called a *plaintext*, into another message, called a *ciphertext*, using some secret key. If the cryptosystem is secure, then the ciphertext can safely be made public, and no party without knowledge of the secret key can recover the plaintext.

Definition 1.3.1. *([98], Definition 1.1) A cryptosystem is a five-tuple ($\mathcal{P}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{E}$, $\mathcal{D}$), where to the following conditions are satisfied:*

1. *$\mathcal{P}$ is a finite set of possible plaintexts;*
2. *$\mathcal{C}$ is a finite set of ciphertexts;*
3. *$\mathcal{K}$ is a finite set of possible keys;*
4. *For each $K \in \mathcal{K}$, there is an encryption rule $e_K \in \mathcal{E}$ and a corresponding decryption rule $d_K \in \mathcal{D}$. Each $e_K : \mathcal{P} \to \mathcal{C}$ and $d_K : \mathcal{C} \to \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext $x \in \mathcal{P}$.*

If the decryption rule d_K, is the same as the encryption rule e_K, or can be easily derived from e_K, then such a cryptosystem is called a *symmetric-key cryptosystem*, as both encryption and decryption can be performed using only knowledge of e_K. In the symmetric-key setting, the value of e_K must be kept secret, otherwise an adversary could decrypt messages. If it is computationally infeasible to determine d_K from e_K, then such a cryptosystem is called a *public-key cryptosystem*, as e_K can be safely made public without allowing an adversary to decrypt messages. By computationally infeasible, it is meant that a computationally-bounded adversary has only a *negligible* chance of succeeding in calculating d_K given e_K, with respect to some pre-defined security parameter ϵ. In many settings, ϵ is taken to be the bit-length of the public parameters of the cryptosystem.

Definition 1.3.2. *A function $\nu : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for any non-zero polynomial $p \in \mathbb{R}[x]$ there exists $m \in \mathbb{N}$ such that for all $n > m$,*

$$|\nu(n)| < \frac{1}{|p(n)|}.$$

In the public-key setting, all the information necessary to apply the encryption rule will be called the *public key*, and the information necessary to apply the decryption rule will be called the *private key* or *secret key*. In the symmetric-key setting, the necessary information to apply e_K (and hence d_K) is the secret key.

Public-key cryptography was first proposed by Diffie and Hellman [43], eventually leading to the discovery of the RSA cryptosystem [88] by Rivest, Shamir, and Adleman, which remains as one of the most important public-key cryptosystems discovered.

Cryptosystem 1.3.3. *The RSA Cryptosystem [88]*

Let p and q be two primes, and define $n = pq$. Also, let $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$, and define

$$\mathcal{K} = \{(n, p, q, d, e) \mid de \equiv 1 \mod \phi(n)\}$$

where ϕ is the Euler totient function. For $\mathcal{K} = (n, p, q, d, e)$, define

$$e_K(x) = x^e \mod n$$

and

$$d_K(y) = y^d \mod n$$

for $x \in \mathcal{M}$ and $y \in \mathcal{C}$. The tuple (e, n) is made public, and the tuple (p, q, d) is kept private.

The RSA cryptosystem works because the values e and d are inverses modulo $\phi(n)$. It is easy to see that Property 4 of Definition 1.3.1 is satisfied:

$$\begin{aligned} d_K(e_K(x)) &= (x^e)^d \mod n \\ &= x^{ed} \mod n \\ &= x^{ed \mod \phi(n)} \mod n \qquad \text{(by Fermat's little theorem)} \\ &= x^1 \\ &= x. \end{aligned}$$

It remains to be seen that knowledge of e_K does not allow an adversary to determine d_K.

Consider an adversary with knowledge of (e, n). In order to apply the decryption rule d_K, the adversary apparently must discover the value of d, the unique integer such that $ed \equiv 1 \mod \phi(n)$. This is easy to compute using the Extended Euclidean Algorithm, assuming the value $\phi(n) = (p-1)(q-1)$ is known; however, because p and q are kept private, the adversary must first factor n to determine p and q. It is on this fact that the security of d_K relies: For sufficiently large n, a computationally-bounded adversary has only a negligible chance of factoring n. Formally, the security of the RSA cryptosystem is parameterized by the bit-length of n. This is not a rigorous proof that an adversary cannot derive the secret key from the public key, but it outlines the basis for security of the private key in RSA.

It is possible that an adversary would be satisfied with something less than learning the private key, and thus gaining the ability to decrypt all messages. Note that in the basic RSA cryptosystem, if two messages $m_1 = m_2$ are encrypted, then the resulting ciphertexts c_1 and c_2 are identical; i.e., encryption is a *deterministic* process. If the adversary observes an encrypted message c, it can easily be checked if c is an encryption of m by checking if $e_K(m) = c$. An adversary could construct a database containing encryptions of all English words and common phrases, potentially allowing for many observed messages to be revealed. To protect against this sort of attack, it is often required that ciphertexts produced by a cryptosystem be *indistinguishable*. By this, it is meant that an adversary, given a single encryption of a known message chosen from the set $\{m_0, m_1\}$, should not be able to determine which message the ciphertext represents with probability significantly greater than $\frac{1}{2}$. This is formalized in Definition 1.3.4.

Definition 1.3.4. *A cryptosystem is said to be indistinguishable under chosen plaintext attack, or IND-CPA, if a probabilistic polynomially bounded adversary cannot win the following game with probability greater than* $\frac{1}{2} + \nu(\epsilon)$*, where* ϵ *is a pre-defined security parameter and* ν *is a negligible function:*

1. *The challenger creates an instance of the cryptosystem using security parameter* ϵ*, and sends the public key to the adversary.*

2. *The adversary may perform a polynomially-bounded number of encryptions or other calculations.*

3. *The adversary chooses two different messages* $m_0, m_1 \in \mathcal{P}$ *and sends them to the challenger.*

4. *The challenger randomly chooses a bit* $b \in \{0, 1\}$ *and sends* $e_K(m_b)$ *to the adversary.*

5. *The adversary may perform a polynomially-bounded number of encryptions or other calculations, then responds with either* 0 *or* 1.

The adversary wins the game if the bit-value chosen by the adversary is the same value chosen by the challenger.

The term *polynomially secure* is sometimes used instead of IND-CPA. Note that the RSA scheme can be made IND-CPA secure through the use of random padding techniques, such as optimal asymmetric encryption padding (OAEP) [7].

Some cryptosystems, called *probabilistic cryptosystems* are designed to incorporate randomness directly into the encryption rule. In probabilistic cryptosystems, the encryption rule is a function of both the public key and some randomly chosen parameter. When RSA is used in conjunction with OAEP, it becomes a probabilistic cryptosystem.

Definition 1.3.5. *([98], Definition 8.1) A probabilistic public-key cryptosystem is defined to be a six-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ where $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{K}$, are defined as in Definition 1.3.1, and $\mathcal{R}$ is a set of randomizers. In addition to the basic properties of a cryptosystem, the following properties should be satisfied:*

1. *Each encryption rule $e_K : \mathcal{P} \times \mathcal{R} \rightarrow \mathcal{C}$ and corresponding decryption rule $d_K : \mathcal{C} \rightarrow \mathcal{P}$ are functions such that*

$$d_K(e_K(b, r)) = b$$

for every plaintext $b \in \mathcal{P}$ and every $r \in \mathcal{R}$.

2. *Let ϵ be a specified security parameter. For any fixed $K \in \mathcal{K}$ and for any $x \in \mathcal{P}$, define a probability distribution $p_{K,x}$ on $\mathcal{C}$ where $p_{K,x}(y)$ denotes the probability that y is the ciphertext given that K is the key over all choices of $r \in \mathcal{R}$. Suppose $x, x' \in \mathcal{P}$, $x \neq x'$, and $K \in \mathcal{K}$. Then all probability distributions $p_{K,x}$ and $p_{K,x'}$ are ϵ-indistinguishable in polynomial time.*

The notion of a probabilistic cryptosystem was first put forward by Goldwasser and Micali [61], who also introduced the concept of *semantic security.* Property 2 of Definition 1.3.5 states the formal requirements for semantic security. Informally, a cryptosystem is semantically secure if, given a ciphertext, the information an adversary can determine about the plaintext is the same as the information that can be determined without the ciphertext. Goldwasser and Micali later showed that semantic security was equivalent to IND-CPA security, and gave the first example of a probabilistic cryptosystem [62], referred to as the Goldwasser-Micali, or GM cryptosystem. The GM cryptosystem is presented in Section 3.2.2.

Indistinguishability under chosen plaintext attack is usually the weakest form of security expected from a cryptosystem. Indeed, the nature of public-key cryptosystems allows the adversary to encrypt a polynomially-bounded number of arbitrary messages without access to any special hardware or non-public information. In many situations it is assumed that the adversary has the ability to perform more complicated actions than choosing a plaintext and calculating the corresponding ciphertext, such as the ability to choose a ciphertext and see the corresponding plaintext. This type of attack is called a *chosen ciphertext attack*. In order to formalize this idea, the idea of a *decryption oracle* is necessary, which acts as a black box that accepts as input a ciphertext, and responds with the corresponding ciphertext in constant time.

Definition 1.3.6. *A cryptosystem is said to be indistinguishable under non-adaptive chosen ciphertext attack, or IND-CCA1 secure, if a probabilistic polynomially bounded adversary cannot win the following game with probability greater than* $\frac{1}{2}+\nu(\epsilon)$ *where* ϵ *is a pre-defined security parameter and* ν *is a negligible function:*

1. *The challenger creates an instance of the cryptosystem using security parameter* ϵ*, and sends the public key to the adversary.*
2. *The adversary is given access to a decryption oracle and may perform a polynomially-bounded number of encryptions, decryptions, or other calculations.*
3. *The adversary chooses two different messages* $m_0, m_1 \in \mathcal{P}$ *and sends them to the challenger.*
4. *The challenger randomly chooses a bit* $b \in \{0,1\}$ *and sends* $e_K(m_b)$ *to the adversary.*
5. *Access to the decryption oracle is suspended. The adversary may perform a polynomially-bounded number of encryptions or other calculations, then responds with either* 0 *or* 1*.*

The adversary wins the game if the bit-value chosen by the adversary is the same value chosen by the challenger.

A non-adaptive chosen ciphertext attack is also called *indifferent chosen ciphertext attack*, or a *lunchtime attack*, so named because an adversary could sneak onto a protected system while its owner was out for lunch (step 2 in Definition 1.3.6), but cannot rely on access at a later date. The strongest form of indistinguishability assumes that the adversary retains access to the decryption oracle after the challenge ciphertext is received, but that the decryption oracle will not respond if the adversary queries the challenge ciphertext.

Definition 1.3.7. *A cryptosystem is said to be indistinguishable under adaptive chosen ciphertext attack, or IND-CCA2, if a probabilistic polynomially bounded adversary cannot win the following game with probability greater than* $\frac{1}{2} + \nu(\epsilon)$ *where* ϵ *is a pre-defined security parameter and* ν *is a negligible function:*

1. *The challenger creates an instance of the cryptosystem using security parameter* ϵ*, and sends the public key to the adversary.*

2. *The adversary is given access to a decryption oracle and may perform a polynomially bounded number of encryptions, decryptions, or other calculations.*

3. *The adversary chooses two different messages $m_0, m_1 \in \mathcal{P}$ and sends them to the challenger.*

4. *The challenger randomly chooses a bit $b \in \{0, 1\}$ and sends $c = e_K(m_b)$ to the adversary.*

5. *The decryption oracle is altered such that it will not respond to a query for c, but will respond for any other valid ciphertext. The adversary may perform a polynomially bounded number of encryptions, decryptions or other calculations, then responds with either 0 or 1.*

The adversary wins the game if the bit-value chosen by the adversary is the same value chosen by the challenger.

It should be clear from the definitions that IND-CCA2 implies IND-CCA1, which in turn implies IND-CPA.

In the chosen plaintext and chosen ciphertext attacks considered so far, a cryptosystem has been considered secure if an adversary cannot distinguish between encryptions of two known messages; however, there may exist situations in which an adversary can modify a ciphertext such that it affects the plaintext in a deterministic way. Cryptosystems which allow this behavior are said to be *malleable*, while cryptosystems that resist this behavior are said to be *non-malleable*.

Definition 1.3.8. *A cryptosystem is said to be malleable if, given a ciphertext c representing plaintext m, it is feasible to compute functions f and g, where f is not the identity function, such that $f(c)$ decrypts to $g(m)$. If it is not feasible to calculate any such f and g, the cryptosystem is said to be non-malleable.*

Clearly, any malleable cryptosystem is not IND-CCA2, as the adversary, given challenge $e_K(m_b)$, can simply send $f(m_b)$ to the decryption oracle to receive $g(m_b)$. The adversary can then calculate $g^{-1}(g(m_b)) = m_b$ to determine which value $b \in \{0, 1\}$ the challenger chose.

Recall the RSA cryptosystem (Cryptosystem 1.3.3). Given the public key $K = (e, n)$ the encryption of two messages m_1 and m_2 can be calculated by $e_K(m_1) = {m_1}^e \mod n$ and $e_K(m_2) = {m_2}^e \mod n$. Taking the product of these two ciphertexts results in

$$\begin{aligned} e_K(m_1)e_K(m_2) \mod n &= {m_1}^e {m_2}^e \mod n \\ &= (m_1 m_2)^e \mod n \\ &= e_K(m_1 m_2). \end{aligned}$$

Thus, the basic RSA cryptosystem is malleable, as the encrypted product of two ciphertexts can be calculated from public information, and hence, the RSA cryptosystem is not IND-CCA2.

Although malleability limits the theoretical security of a cryptosystem, it is not necessarily an undesired property. A special class of cryptosystems, called *homomorphic cryptosystems*, are designed specifically to allow simple calculations on ciphertexts. In most cases, homomorphic cryptosystems allow someone to take two encrypted messages $e_K(m_1)$ and $e_K(m_2)$ and calculate $e_K(m_1 + m_2)$ or $e_K(m_1 m_2)$ without knowledge of the private key. A detailed discussion of homomorphic cryptosystems is given in the next Chapter.

Chapter 2

Overview of Homomorphic Cryptography

As was shown in the previous section, the basic RSA cryptosystem is malleable; that is, it is possible to alter the plaintext that a ciphertext represents in a deterministic way. By taking the product of two messages encrypted under the same public key, the result is a ciphertext that will decrypt to the product of the two messages. While it is possible to envision a scenario where an adversary could take advantage of malleability in a cryptosystem, such as altering an encrypted bank deposit, it is also possible to use malleability in a beneficial manner. For example, suppose Bob has knowledge of some secret function f, and Alice possesses a value secret x for which she wishes to learn $f(x)$. By using an appropriate malleable cryptosystem it may be possible for Alice to send $e_K(x)$ to Bob, who then calculates the encrypted result $e_K(f(x))$ and returns it to Alice, who now learns the value $f(x)$ without learning f.

2.1 Privacy Homomorphisms

The idea of performing simple computations on encrypted messages was first put forward by Rivest, Adleman, and Dertouzous [86], who referred to such computations as *privacy homomorphisms*. The original motivation for privacy homomorphisms was to allow for an encrypted database to be stored by an untrusted third party, while still allowing to owner to perform simple updates and queries such that nothing about the database contents is revealed to the third party.

Definition 2.1.1. *A privacy homomorphism is a homomorphism ϕ from an algebraic system U consisting of a set $\mathcal{S}$, some operations $f_1, f_2, \ldots$, some predicates $p_1, p_2, \ldots$, and some distinguished constants $s_1, s_2, \ldots$, to an algebraic system C consisting of a set $\mathcal{S}'$, some operations $f'_1, f'_2, \ldots$, some predicates $p'_1, p'_2, \ldots$, and some distinguished constants $s'_1, s'_2, \ldots$ such that:*

1. $(\forall i)(a,b,c,\ldots) \in S'\ [f'_i(a,b,\ldots) = c \Rightarrow f_i(\phi(a),\phi(b),\ldots) = \phi(c)]$

2. $(\forall i)(a,b,c,\ldots) \in S'\ p'(a,b,\ldots) \equiv p(\phi(a),\phi(b),\ldots)$

3. $(\forall i)\ \phi(s'_i) = s_i$.

The function ϕ is called the decoding function, and the function ϕ^{-1} is called the encoding function.

In this setting, ϕ serves as an decryption key, while the algebraic system U is kept private, and the algebraic system C is made public. As an example, assume a user wishes to calculate $f_1(d_1,d_2)$ from a set of values stored by a third party, without revealing f_1. The user submits a request for $f'_1(\phi^{-1}(d_1),\phi^{-1}(d_2))$, and, upon receiving the response, calculates

$$\begin{aligned}\phi(f'_1(\phi^{-1}(d_1),\phi^{-1}(d_2))) &= f_1(\phi(\phi^{-1}(d_1)),\phi(\phi^{-1}(d_2)))\ \text{(By Property 1)}\\ &= f_1(d_1,d_2).\end{aligned}$$

In addition to the mathematical properties of ϕ, Rivest et al. also propose that a privacy homomorphism must satisfy the following properties:

1. ϕ and ϕ^{-1} should be easy to compute.
2. The operations f'_i and predicates p'_i in C should be easy to compute.
3. $\phi^{-1}(d_i)$ should not require much more space than d_i.
4. Knowledge of $\phi^{-1}(d_i)$ for many values of d_i should not be sufficient to reveal ϕ.
5. Knowledge of d_i and $\phi^{-1}(d_i)$ for several values of d_i should not reveal ϕ.
6. The operations and predicates in C should not be sufficient to yield an efficient computation of ϕ.

Conditions 4 and 5 are equivalent to chosen ciphertext and chosen plaintext attacks, respectively. Condition 6 is important when dealing with comparison predicates such as the $\leq$ predicate. It has been observed that if an adversary can calculate the encoding of arbitrary constants and a predicate for total order, such as $\leq$, is available, then there is no secure privacy homomorphism from U to C; that is, an adversary can determine $\phi(a)$ from a. This follows from the adversary's ability to calculate $\leq$ on $2^1, 2^2, 2^3, \ldots$ until an upper bound is located, and then perform a binary search for the encoded value.

Rivest et al. proposed several examples of privacy homomorphisms, with multiplication under RSA being one of the examples. These examples were intended as "proof of concepts" rather than robust systems, and it was later shown by Brickell and Yacobi that each of the proposed systems, except the RSA based system, were vulnerable to chosen plaintext or chosen ciphertext attacks [17].

2.2 Homomorphic Cryptosystems in an Abstract Setting

Most cryptosystems are defined over algebraic groups or rings, such as $\mathbb{Z}_n$ or $\mathbb{Z}_n^*$, which can be considered algebraic systems as in Definition 2.1.1. Cryptosystems defined over a group naturally support a single operation, usually denoted by multiplication or addition for cryptographic purposes, and cryptosystems defined over a ring naturally support two operations, usually denoted by addition and multiplication. Thus, if the encryption rule for a cryptosystem, where both the plaintext space and the message space are groups (or rings), is a homomorphism, then such a cryptosystem is a privacy homomorphism. Such systems are referred to as *homomorphic cryptosystems*, as they are usually general-purpose cryptosystems that also provide limited privacy homomorphism capabilities.

Definition 2.2.1. *A homomorphic cryptosystem is a cryptosystem where the set of possible plaintexts $\mathcal{P}$ and the set of possible ciphertexts $\mathcal{C}$ are both groups such that for any $K \in \mathcal{K}$ and any two ciphertexts $c_1 = e_K(m_1)$, $c_2 = e_K(m_2)$, the following condition holds:*

$$d_K(c_1 \cdot c_2) = m_1 \cdot m_2$$

where $\cdot$ represents the respective group operations in $\mathcal{C}$ and $\mathcal{M}$.

This definition extends naturally to a cryptosystem defined over a ring.

Definition 2.2.2. *An algebraically homomorphic cryptosystem, or ring homomorphic cryptosystem, is a cryptosystem where the set of possible plaintexts $\mathcal{P}$ and the set of possible ciphertexts $\mathcal{C}$ are both rings such that for any $K \in \mathcal{K}$ and any two ciphertexts $c_1 = e_K(m_1)$, $c_2 = e_K(m_2)$, the following conditions hold:*

1. $d_K(c_1 + c_2) = m_1 + m_2$
2. $d_K(c_1 \cdot c_2) = m_1 \cdot m_2$

where $+$ and $\cdot$ represent the respective ring operations in $\mathcal{C}$ and $\mathcal{M}$.

A survey of known homomorphic cryptosystems is given in Section 3.2.

For now, a homomorphic cryptosystem can be thought of as an oracle, or black box, that, when given two ciphertexts and an operation, returns an encryption of the result of that operation on the two corresponding plaintexts. Because addition and multiplication are common operations provided by a homomorphic cryptosystem, the symbol $\boxplus$ will be used with ciphertexts to denote the operation which results in an encryption of the sum of the two plaintext messages, and the symbol $\boxtimes$ will be used to denote the operation on ciphertexts needed to produce an encryption of the product of two plaintext messages:

$$\begin{aligned} d_K(e_K(m_1) \boxplus e_K(m_2)) &= m_1 + m_2 \\ d_K(e_K(m_1) \boxtimes e_K(m_2)) &= m_1 m_2. \end{aligned}$$

In general, $\boxplus$ and $\boxtimes$ may be any probabilistic polynomial time algorithms which achieve the desired result.

2.3 Security of Homomorphic Cryptosystems

Homomorphic cryptosystems have the useful property of allowing anybody to perform simple computations on encrypted messages. These cryptosystems can be used for many different privacy-preserving protocols, with several examples described in Chapter 5; however, before using them in protocols their theoretical limits must be considered. The notions of security defined in Section 1.3 assumed the adversary had the ability to encrypt and decrypt messages in various settings, but did not account for the ability of the adversary to perform homomorphic operations on ciphertexts. The security of cryptosystems when homomorphic operations are possible, and what sort of attacks are introduced by homomorphic operations are considered in this section.

2.3.1 Theoretical Limits of Homomorphic Cryptosystems

It should be immediately obvious that no homomorphic cryptosystem can be IND-CCA2 secure. This follows from the fact that any homomorphic cryptosystem is malleable.

Theorem 2.3.1. *If a cryptosystem $\mathcal{C}$ is homomorphic (or algebraically homomorphic), then it is not IND-CCA2 secure.*

Proof. Consider the game defined in Definition 1.3.7. In step 5, the adversary has chosen two messages m_0 and m_1, and has received a challenge $e_K(m_b)$ from which it must be determined if $b = 0$ or $b = 1$. The adversary may request an encryption of some known constant c, and then use the homomorphic properties of the cryptosystem to calculate $e_K(m_b) \boxplus e_K(c) = e_K(m_b + c)$. The adversary then sends a decryption query to the oracle to learn $m_b + c$, and can easily determine which of m_0 or m_1 is the challenge. $\square$

In light of this negative result, Yu, Leiwo, and Premkumar [104] investigated the theoretical limits of ciphertext indistinguishability for homomorphic cryptosystems. Their analysis was done in the black box model, which, as mentioned in Section 2.2, assumes that the operations $\boxplus$ and $\boxtimes$ are implemented by a black box that, given two ciphertexts and an operation, returns an encryption of the result of that operation on the two corresponding plaintexts. Although this assumption might seem unrealistic, it can be easily modeled in a real world setting through the use of tamper-proof hardware that can decrypt messages, perform simple arithmetic, and then encrypt the result.

Knowing that IND-CCA2 security is impossible to achieve, Yu et al. considered the next strongest form of indistinguishability, IND-CCA1. For a cryptosystem to be IND-CCA1 secure, an adversary must gain only a negligible advantage in distinguishing messages after being given polynomially-many queries to a decryption oracle. Yu et al. have shown that if a (non-homomorphic) cryptosystem is IND-CCA1, then its black box implementation as an algebraically homomorphic cryptosystem is also IND-CCA1 secure.

Recall the game from Definition 1.3.6. Let $\mathcal{O}^{e,d}$ be an oracle that answers both encryption and decryption queries, and let $\mathcal{O}^{d}$ be an oracle that only answers decryption queries. In step 2 of the game, the adversary is given access to $\mathcal{O}^{e,d}$, and in step 5 it is given access to $\mathcal{O}^{e}$. In order to show that the algebraically homomorphic black box implementation of an IND-CCA1 cryptosystem $\mathcal{C}$ is also IND-CCA1 secure, the oracles in step 2 and step 5 are replaced with $\mathcal{O}^{e,d,\boxplus,\boxtimes}$ and $\mathcal{O}^{e,\boxplus,\boxtimes}$ respectively. These oracles are identical to the non-black box oracles, but will also answer queries requesting a homomorphic operation on two ciphertexts.

Theorem 2.3.2. *([104], Theorem 1) Let $\mathcal{C}$ be an IND-CCA1 secure cryptosystem (with respect to security parameter ϵ) such that the homomorphic operations $\boxplus$ and $\boxtimes$ are computable on ciphertexts in $\mathcal{C}$. Then the algebraically homomorphic black box implementation of $\mathcal{C}$ is also IND-CCA1 secure.*

Proof. The theorem holds if replacing the oracles $\mathcal{O}^{e,d}$ and $\mathcal{O}^{e}$ in Definition 1.3.6 with the $\mathcal{O}^{e,d,\boxplus,\boxtimes}$ and $\mathcal{O}^{e,\boxplus,\boxtimes}$ does not give the adversary a sufficient advantage to win the game with more than $\frac{1}{2}+\nu(\epsilon)$ probability, where ν is a negligible function. In step 2, consider a request for $\boxplus$ or $\boxtimes$ made to $\mathcal{O}^{e,d,\boxplus,\boxtimes}$. The same request could be simulated with access to $\mathcal{O}^{e,d}$ by decrypting both messages, manually performing the desired operation, and then submitting the result as an encryption query. Thus, the adversary does not gain any additional information in step 2. Next, consider step 5 of the game. Access to $\mathcal{O}^{e}$ is not sufficient to simulate $\mathcal{O}^{e,d,\boxplus,\boxtimes}$. Instead, it can be shown that the information returned by $\mathcal{O}^{e}$ is computationally indistinguishable from the output of $\mathcal{O}^{e,\boxplus,\boxtimes}$. Assume that the adversary replaces $\mathcal{O}^{e,d,\boxplus,\boxtimes}$ with the oracle $\mathcal{O}^{e,+,\times}$, which, given a query on $(e_K(m_1),\ e_K(m_2),\ \{+,\times\})$ returns $e_K(0)$. This oracle is clearly simulatable with access only to $\mathcal{O}^{e}$. Because $\mathcal{C}$ is an IND-CCA1 secure cryptosystem, the adversary cannot distinguish between $e_K(0)$, $e_K(m_1+m_2)$, and $e_K(m_1m_2)$. Hence, the adversary's advantage in distinguishing between queries made to $\mathcal{O}^{e,\boxplus,\boxtimes}$ and $\mathcal{O}^{e,+,\times}$ is negligible, which implies the advantage gained by the adversary with access to $\mathcal{O}^{e,d,\boxplus,\boxtimes}$ is negligible. Since the adversary gains no non-negligible information in both step 2 and step 5, the adversary cannot win the game with access to $\mathcal{O}^{e,d,\boxplus,\boxtimes}$ and $\mathcal{O}^{e,\boxplus,\boxtimes}$ with more than $\frac{1}{2}+\nu(\epsilon)$ probability, and the algebraically homomorphic black box implementation of $\mathcal{C}$ is IND-CCA1 secure. $\square$

It should be obvious that Theorem 2.3.2 would also hold if $\mathcal{C}$ were a non-algebraically homomorphic cryptosystem. Thus, in light of Theorems 2.3.1 and 2.3.2, the following theorem can be stated.

Theorem 2.3.3. *IND-CCA1 is a tight upper bound for the security of a homomorphic (or algebraically homomorphic) cryptosystem.*

Proof. The theorem follows directly from Theorems 2.3.1 and 2.3.2. □

Although Theorem 2.3.2 shows that IND-CCA1 security is achievable, its proof relies on the fact that the homomorphic operations are implemented in an ideal black box model. In practice, $\boxplus$ and $\boxtimes$ are probabilistic polynomial time algorithms, not black box oracles. The question of whether or not IND-CCA1 security is achievable in an algorithmic setting is still an open problem. The impossibility of IND-CCA2 security for a homomorphic cryptosystem does not rely on the black box model, so it can still be stated that IND-CCA1 is an upper bound for an algorithm-based homomorphic cryptosystem, but possibly not a tight upper bound.

2.3.2 Known Attacks Against Homomorphic Cryptosystems

In some cases, homomorphic operations allow an adversary to break a homomorphic cryptosystem by allowing an encrypted message to be recovered without knowledge of the secret key. Some examples of such attacks are presented in this section.

Ahituv, Lapid, and Neumann [4] have demonstrated a chosen-ciphertext attack against a homomorphic cryptosystem where the ciphertext operation for $\boxplus$ is straightforward addition. Let e_K be an encryption function which maps an n-bit message to an m-bit message such that $d_K(e_K(m_1) + e_K(m_2)) = m_1 + m_2$, and consider the following messages:

$$\begin{array}{rcl} e_K(a_1) & = & [1, 0, 0, \ldots, 0] \\ e_K(a_2) & = & [0, 1, 0, \ldots, 0] \\ & \vdots & \\ e_K(a_m) & = & [0, 0, \ldots, 0, 1]. \end{array}$$

If an adversary is able to learn the plaintext message a_i for the ciphertext consisting of a 1-bit at position i and 0-bits elsewhere, then the adversary can efficiently decrypt any ciphertext. Let $c = e_K(m)$ for some message m, and let c_i be the bit-value of c at position i. The adversary can retrieve m by calculating

$$m = \sum_{i=1}^{m} c_i a_i.$$

This attack works because the chosen ciphertexts form a basis for which it is trivial to compute the representation of any other ciphertext. Because the corresponding plaintexts are known for each basis element, and the same addition operation works in each domain, the plaintext message is easily recovered. Yu, Leiwo, and Premkumar [104] have observed that this attack works for any system in which $\boxplus$

is a linear operation, such as addition. Additionally, they have observed that this attack has been incorrectly interpreted by some as stating that no additive homomorphic cryptosystem can be secure, a statement which is incorrect, as the attack fails when $\boxplus$ is a non-linear operation, such as multiplication. This attack is also sometimes incorrectly referred to as a chosen-plaintext attack [46, 104], despite the fact that the attack adversary is required to choose m specific ciphertexts.

Boneh and Lipton [15] have shown that any algebraically homomorphic cryptosystem can be broken in sub-exponential time. Their proof is based on the *black box field problem* (BBFP) and the fact that it can be solved in sub-exponential time.

Definition 2.3.4. *([15], Definition 1) A black box field is a six-tuple:* $(p,\ n,\ h,\ F,\ G,\ T)$ *where* p *is a prime and* n *is a positive integer representing the encoding length. The functions* h, F, G, T *are defined as follows:*

1. *The function* $h : \{0,1\}^n \to \mathbb{F}_p$ *associates a field element with every* n*-bit binary string. The function* h *is surjective.*
2. *The functions* $F, G : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *perform addition and multiplication. They satisfy the following relations:* $h(F(x,y)) = h(x) + h(y)$ *and* $h(G(x,y)) = h(x)h(y)$*.*
3. *The function* $T : \{0,1\}^n \times \{true, false\}$ *tests equality of two black-box elements:* $T(x,y) = true$ *if and only if* $h(x) = h(y)$*.*

Note that the function h, which maps binary strings to field elements, is surjective, but not injective. Because many different binary strings can be mapped to the same field element the notation $[x]$ is used to refer to some binary string such that $h([x]) = x$.

Definition 2.3.4 appears very similar to Definition 2.2.2. The function h acts like a decryption function which takes a binary message and maps it to a unique element of $\mathbb{F}_p$, and the functions F and G act like the operations $\boxplus$ and $\boxtimes$. Because there is a potentially many-to-one mapping of binary strings to field elements, the definition also seems to fit that of a probabilistic cryptosystem (Definition 1.3.5), although the function T acts like an oracle that can distinguish "ciphertexts" in a black box field setting.

Although black box fields appear very similar to algebraically homomorphic cryptosystems, Definition 2.3.4 does not state anything about the difficulty of determining x from $[x]$ without knowledge of h, a task which should be difficult if black box fields are used to model algebraically homomorphic cryptosystems. Boneh and Lipton address this by proposing the black box field problem.

Definition 2.3.5. *([15], Definition 2) Let* (p, n, h, F, G, T) *be a black box field for some prime* p*. Denote the map sending* x *to some* $[x]$ *by* $[]$*. The black box field problem (BBFP)is the following: Find an algorithm* A *that, given* p *and oracles for*

$F, G, T, []$ *and an element* $[\alpha] \in \mathbb{F}_p$*, finds* α *explicitly. Formally,* $A^{F,G,T,[]}([\alpha]) = a$ *where* $a \equiv \alpha \mod p$*. The algorithm is said to run in polynomial time if it runs in time* $\log^{O(1)} p$*. The algorithm is sub-exponential if it runs in sub-exponential time in* $\log p$*.*

Boneh and Lipton then show that a sub-exponential solution to BBFP implies a sub-exponential algorithm to break an algebraically homomorphic cryptosystem; i.e., given $e_K(m)$ there exists a sub-exponential algorithm to determine m.

Theorem 2.3.6. *([15], Theorem 5) Suppose that BBFP in a finite field of size p can be solved in time* $T_{BBF}(p)$*. Then any algebraically homomorphic cryptosystem* (d_K, e_K) *over a plaintext ring of size n can be broken in expected time*

$$O\left(T_{BBF}(n) + \exp\left((1 + o(1))\sqrt[3]{\log n \log^2 \log n}\right)\right)$$

The proof of this theorem is based on factoring $n = \prod p_i$ into prime factors and solving BBFP over $\mathbb{F}_{p_i}$ for each p_i. Given $x = e_K(m)$, a black box field (p_i, n, h, F, G, T) is created for each p_i, with $h(x) = d_K(m) \mod p_i$, $F = \boxplus$, and $G = \boxtimes$. A solution to BBFP for each of these fields yields $m \mod p_i$ for each p_i, allowing m to be recovered through Chinese remaindering.

Boneh and Lipton then go on to show that BBFP does indeed have a sub-exponential solution, thus proving that any algebraically homomorphic cryptosystem defined over a ring of size n can be broken in sub-exponential time. Their result depends on the smoothness assumption, which is stated in [15] as follows: Let $L_\alpha(p) = \exp\left(\log^\alpha p \log\log^{1-\alpha} p\right)$ and let $d(x)$ be the largest prime divisor of x. The smoothness assumption is the assumption that integers chosen uniformly in the range $[\,p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}\,]$ satisfy

$$\Pr_x[d(x) < L_\alpha(p)] > \frac{1}{L_{1-\alpha}(p)^{1-\alpha+o(1)}},$$

which is known to be true when x is in the range $[1, p]$.

Theorem 2.3.7. *([15], Theorem 8) Let K be a finite field of size p given as a black box field. Under the smoothness assumption, BBFP can be solved using* $O(\log p)$ *space and expected time*

$$L_{\frac{1}{2}}(p)^{2+o(1)} = \exp\left((2 + o(1))\sqrt{\log p \log\log p}\right).$$

Although this result does limit the theoretical security of algebraically homomorphic cryptosystems, it should not be interpreted as stating that a secure algebraically homomorphic cryptosystem cannot exist. Many cryptosystems, such as RSA, rely on the difficulty of factoring $n = pq$ where p and q are large primes; however, sub-exponential algorithms for factoring n exist, such as the elliptic curve method [70]. Despite this fact, the RSA cryptosystem remains as one of the most widely used public-key cryptosystems. Additionally, because BBFP requires functions for both addition and multiplication, this result does not hold for a non-algebraically homomorphic cryptosystem.

In addition to the attacks presented thus far, there exist quantum algorithms that can break algebraically homomorphic cryptosystems more efficiently. Shor's algorithm [93] can be used to efficiently factor integers, thus breaking any scheme which relies on the difficulty of factoring large integers for security. The period finding sub-routine of Shor's algorithm can also be used for attacks specifically on algebraically homomorphic cryptosystems. Additionally, a different quantum attack against algebraically homomorphic cryptosystems has been given by van Dam, Hallgren, and Ip [101].

2.3.3 Alternate Security Notions for Homomorphic Cryptosystems

IND-CCA2 is often considered to be an unreasonably strong security assumption. Consider an IND-CCA2 cryptosystem C and define C' such that $e_K{}'(m) = e_K(m)||b$ where b is a random bit, and e_K is the encryption method from C. The decryption of messages in C' can be calculated by throwing away the least significant bit and using d_K from C. Despite the fact that C' clearly provides just as much message security as C, it is not IND-CCA2 secure. An adversary can simply flip the least significant bit of a challenge ciphertext and submit the new ciphertext to the decryption oracle to win the game defined in Definition 1.3.7. The "loss of security" provided by C' stems from the fact that C has been changed into a malleable cryptosystem, even though this malleability provides no additional information to an adversary. In this section, two weakenings of IND-CCA2 are presented (neither of which are suitable for homomorphic cryptosystems), followed by a generalization of these two weakenings that also provides a notion of adaptive chosen ciphertext indistinguishability for homomorphic cryptosystems.

Shoup [96] has proposed that adaptive chosen ciphertext security be altered to allow for *benign malleability.* If a cryptosystem allows for malleable operations that are easy to detect and do not otherwise grant an adaptive adversary any non-negligible advantage in distinguishing ciphertexts, i.e., the malleable operation is benign, then it should be possible to formally state that such a system provides more than just IND-CCA1 security. An, Dodis, and Rabin have called this *generalized CCA2 (gCCA2)* security [5].

Definition 2.3.8. *Let R be an equivalence relation such that given two ciphertexts c_1 and c_2:*

- $R(c_1, c_2) = true$ *implies*$d_K(c_1) = d_K(c_2)$
- *R can be calculated efficiently, i.e., there exists a polynomial time algorithm to determine $R(c_1, c_2)$.*

A cryptosystem is said to be benignly malleable, or indistinguishable under generalized adaptive chosen ciphertext attack (IND-gCCA2), if a polynomially bounded adversary cannot win the following game with probability greater than $\frac{1}{2}+\nu(\epsilon)$ *where* ϵ *is a pre-defined security parameter and* ν *is a negligible function:*

1. *The challenger creates an instance of the cryptosystem using security parameter* ϵ*, and sends the public key to the adversary.*
2. *The adversary is given access to a decryption oracle and may perform any number of encryptions, decryptions, or other calculations.*
3. *The adversary chooses two different messages* $m_0, m_1 \in \mathcal{P}$ *and sends them to the challenger.*
4. *The challenger randomly chooses a bit* $b \in \{0, 1\}$ *and sends* $c = e_K(m_b)$ *to the adversary.*
5. *The decryption oracle is altered such that it will not respond to a query* q *if* $R(q, c) =$ *true, but will respond for any other valid ciphertext. The adversary may perform any number of encryptions, decryptions or other calculations, then responds with either* 0 *or* 1*.*

The adversary wins the game if the bit-value chosen by the adversary is the same value chosen by the challenger.

Consider the modified cryptosystem C' defined earlier. The equivalence relation R can be defined to return true if both ciphertexts are identical, except for the least significant bit. Because the original cryptosystem C is IND-CCA2 secure, there does not exists a polynomial time algorithm to distinguish between ciphertexts when two ciphertexts are not related through R, and C' can be declared be IND-gCCA2 secure.

Although Definition 2.3.8 is weaker than Definition 1.3.7, a homomorphic cryptosystem still cannot achieve IND-gCCA2 security. The equivalence relation R only returns true if two ciphertexts decrypt to the same plaintext message; thus, the adversary can still use the oracle to decrypt $e_K(m_b + k)$, for example, to solve for m_b. The malleability provided by homomorphic cryptosystems is not "benign" enough to satisfy IND-gCCA2.

A further weakening of IND-CCA2, called *replayable-CCA2 (rCCA2)* has been proposed by Canetti, Krawczyk, and Nielson [20]. The rCCA2 game is identical to the gCCA2 game in Definition 2.3.8, except the oracle in step 5 is altered so that it will not respond to a decryption request for a ciphertext c if $d_K(c) \in \{m_0, m_1\}$. This weakening is motivated by the fact that from a protocol point of view, the ability to generate a new ciphertext from c that decrypts to the same message does not give an adversary any more advantage than simply replaying the same ciphertext c, which is possible in any IND-CCA2 cryptosystem. Thus, as stated by Krawczyk et al., "RCCA security would have 'essentially the same effect' as CCA security."

Unfortunately, rCCA2 is still too strong for a homomorphic cryptosystem to achieve, for the same reasons that IND-CCA2 and IND-gCCA2 are also too strong. While both gCCA2 and rCCA2 security attempt to allow some form of malleability, both definitions are restricted to transformations that allow an adversary to calculate a new ciphertext c' from an existing ciphertext c such that $d_K(c') = d_K(c)$. Prabhakaran and Rosulek [84, 83] have attempted to address this problem by altering IND-CCA2 such that a cryptosystem may be malleable with respect to some function (or set of functions) $\mathcal{T}$, but non-malleable under all other operations. Under their model, gCCA2 and rCCA2 can be modeled as homomorphic cryptosystems where the only allowable transformation is the identity function. They refer to this as *homomorphic chosen ciphertext (hCCA2)* security with respect to some set of allowable transformations $\mathcal{T}$.

The hCCA2 game requires some additional oracles not found in the original IND-CCA2 game, that allow the adversary to determine if one ciphertext was generated from another through an allowable transformation. Prabhakaran and Rosulek call these as "rigged oracles." Let $RigEnc_{pk}$ be an oracle that, for a public key pk, outputs (ζ, S), where ζ is indistinguishable from a ciphertext encrypted under pk, and S is some auxiliary information used to track transformations. Let $RigExt_{sk}(\zeta', S)$ be an oracle that determines if ζ' was derived from ζ through allowable transformations in $\mathcal{T}$, and, if so, returns the transformation. The value S is used by the oracle to link ζ and ζ' together if they are related. In the hCCA2 game, S is not available to the adversary, so Prabhakaran and Rosulek introduce "guarded" versions of $RigEnc$ and $RigExt$ for the adversary to use. Let $gRigEnc_{pk}$ return ζ_i, where the i'th call to $RigEnc_{pk}$ returns (ζ_i, S_i), and let $gRigExt_{sk}(\zeta, i) = RigExt_{sk}(\zeta, S_i)$.

Definition 2.3.9. *A public-key cryptosystem said to be indistinguishable under adaptive homomorphic chosen ciphertext attack (IND-hCCA2) if a polynomiall bounded adversary cannot win the following game with probability greater than* $\frac{1}{2} + \nu(\epsilon)$ *where* ϵ *is a pre-defined security parameter and* ν *is a negligible function:*

1. *The challenger creates an instance of the cryptosystem using security parameter* ϵ*, and sends the public key to the adversary.*

2. *The adversary is given access to a decryption oracle* Dec_{sk} *and the guarded oracles* $gRigEnc_{ps}$ *and* $gRigExt_{sk}$*, and may perform any number of oracle queries or other calculations.*

3. *The adversary chooses a message* m^* *and sends it to the challenger.*

4. *The challenger chooses a random bit* $b \in \{0, 1\}$*. If* $b = 0$*, set* $\zeta^* = e_K(m^*)$ *and* $RigDec_{sk} = Dec_{sk}$*, otherwise, set* $\zeta^* = RigEnc_{pk}$ *and set*

$$RigDec_{sk}(\zeta) = \begin{cases} T(m^*) & \text{if } RigExt(\zeta^*, S^*) \text{ returns } T \\ Dec_{sk}(\zeta) & \text{otherwise.} \end{cases}$$

   The challenger sends ζ^* *to the adversary.*

5. *Dec_{sk} is replaced with $RigDec_{sk}$ and the adversary may perform any number of oracle queries or other calculations. The adversary then responds with either* 0 *or* 1.

The adversary wins the game if the bit-value chosen by the adversary is the same value chosen by the challenger.

The hCCA2 game follows the same basic structure as the previously defined CCA2 games; however, it has been altered such that the adversary is no longer distinguishing between encryptions of two known messages m_0 and m_1, but between encryptions of a known message m^* and a random message. In step 4, the decryption oracle is altered to respond with $T(m^*)$ if the query ciphertext is related to the challenge ciphertext, and acts as a regular decryption oracle otherwise.

Consider a regular CCA2 secure cryptosystem C under the hCCA2 game and assume that the adversary chooses the challenge m_1, but m_0 is fixed and publicly known. In the regular CCA2 setting, the only allowable operation on ciphertexts is the identity function ID. Define $RigEnc$ to return $(e_K(m_0), e_K(m_0))$ (both identical encryptions) when queried, and $RigExt(c, S)$ returns ID if $c = S$ (and some pre-defined error message otherwise), then the hCCA2 game appears very similar to the CCA2 game. If $b = 0$ and the adversary submits ζ^* to $RigDec_{sk}$, then the adversary receives m_1 in response. Similarly, if $b = 1$ and the adversary submits ζ^* to $RigDec_{sk}$, then the adversary learns $ID(m_1) = m_1$. If the adversary submits a different message to $RigDec_{sk}$, then the adversary simply learns the decryption of that message. Although this is slightly different than the game in Definition 1.3.7, it retains the same level of security. Prabhakaran and Rosulek have also shown that $RigEnc$ and $RigExt$ can be defined to capture gCCA2 and rCCA2 security.

Theorem 2.3.10. *([84], Theorem 1) CCA2, gCCA2, and rCCA2 can be obtained as special cases of hCCA2 by appropriately defining RigEnc and RigExt.*

The proof of this theorem can be found in [84].

It is worth noting that the hCCA2 game is only defined in terms of unary operations, such as multiplication by a constant, or the identity function. Prabhakaran and Rosulek also note that in addition to achieving hCCA2 security, it should not be possible for an adversary to determine whether a message was generated through a series of homomorphic operations, or simply by encrypting a single plaintext message, i.e., a ciphertext generated by another through an allowable transformation should be *unlinkable* to the original. In the hCCA2 game, the rigged oracles provide linkability, but only on messages generated by $RigEnc_{pk}$, which are indistinguishable from ciphertexts generated by encrypting a message under the public key pk. Without these oracles, linkability should require knowledge of the secret key, which is sufficient to win the game. A formal definition for unlinkability with respect to hCCA2 is given in [84].

Although extending the hCCA2 game to allow binary transformations is desirable, Prabhakaran and Rosulek demonstrate that hCCA2 security is impossible to achieve if any of the allowable (binary) transformations are quasigroup operations, i.e., the equation $a * b = c$ is uniquely determined by only two of the three values.

Chapter 3

A Survey of Homomorphic Cryptosystems

This chapter is divided into two main sections. First, a broad overview of the computational problems utilized by homomorphic cryptosystems to achieve security is provided. These descriptions are meant to provide enough information to clearly define what the problem is, and what the best known approaches to solving them are. The second half of the chapter consists of a survey of known homomorphic cryptosystems, generally in the order they were discovered. Many variants of the main homomorphic cryptosystems exist, which are presented alongside the original cryptosystem when the modifications are small, or presented in a new section when the changes are substantial.

Recently, a survey titled "A survey of homomorphic cryptography for non-specialists" has been given by Fontaine and Galand [47], providing an excellent overview of the current state of research on homomorphic cryptography. The survey given here is more comprehensive, with a proper statement of each cryptosystem given alongside the assumptions required for security. Several variants of popular cryptosystems are also discussed, including the necessary modifications to security assumptions. An attempt is also made to provide an outline of a proof of security for each of the major cryptosystems.

3.1 Building Blocks for Homomorphic Cryptosystems

The usual method of demonstrating that a cryptosystem is secure is showing that if an adversary can invert the encryption function, or break semantic security, then the same adversary can break some instance of a problem believed to be intractable. Thus, if the problem is intractable, the cryptosystem should be secure. In this section, several problems are presented upon which homomorphic cryptosystems are built. For each problem presented, for example, the Factoring Problem, the associated assumption, the Factoring Assumption, will be the assumption that the problem is intractable for appropriately chosen parameters.

Definition 3.1.1. *([72], Definition 3.1, 3.2) Let A and B be two computational problems. A is said to polytime reduce, or, be polynomially reducible to B, if there exists an algorithm that solves A, which uses, as a subroutine, a hypothetical algorithm for solving B, and which runs in polynomial time if the algorithm for solving B does.*

If A is polytime reducible to B, and B is polytime reducible to A, then A and B are said to be computationally equivalent.

The hypothetical algorithm used for solving a computational problem is often referred to as an oracle, and works like a black box that, given an instance of the computational problem, returns a solution in polynomial time with non-negligible probability.

3.1.1 The Factoring Problem

The fundamental theorem of arithmetic states that every integer $n \geq 2$ can be expressed uniquely as a product of prime factors, i.e.,

$$n = {p_1}^{e_1} {p_2}^{e_2} \cdots {p_k}^{e_k}$$

where the p_i's are distinct primes. The problem of determining the prime factorization of n, particularly when n is large and has few prime factors (typically two or three), forms the basis of security for many cryptosystems.

Definition 3.1.2. *Given an integer n, the factoring problem is the problem of calculating the p_i's such that*

$$n = {p_1}^{e_1} {p_2}^{e_2} \cdots {p_k}^{e_k}$$

where each p_i is a prime and $p_i \neq p_j$ when $i \neq j$.

Currently, the best known methods for factoring are the number field sieve [69] and the elliptic curve method [70]. In general, the number field sieve is the most efficient algorithm for factoring n, assuming n has no small prime factors. The running time of the number field sieve depends on the bit-length of n, while the elliptic curve method depends on the bit-length of the smallest factor of n. In cryptographic applications, n is usually selected such that each of its prime factors are approximately the same bit-length. While $n = pq$ for primes p and q is commonly used, some cryptosystem use $n = p^2q$. The number field sieve remains the most efficient method of factoring numbers of this form; however, some elliptic curve factoring approaches specific to $n = p^2q$ have been studied, such as the approach given by Peralta and Okamoto [82].

The difficulty of factoring n can be parameterized by ϵ, the bit-length of n, with the assumption that each prime factor of n is approximately the same size. To remain secure against modern attacks, it is generally regarded that ϵ should be at least 1024, if not 2048.

3.1.2 RSA / e'th Root Problem

The RSA cryptosystem was presented earlier as Cryptosystem 1.3.3. The RSA encryption function is $c = m^e \mod n$ where $m \in \mathbb{Z}_n$ and $n = pq$ for distinct odd primes such that the factoring problem is hard on n. Decrypting the ciphertext c is equivalent to taking the e'th root of $c \mod n$. When the factorization of n, the private key, is unknown, this task should be intractable.

Definition 3.1.3. *Let $n = pq$ for distinct odd primes p and q. Given integers $c, e \in \mathbb{Z}_n$, with $e > 2$, the RSA Problem is the problem of finding m such that $c = m^e \mod n$, without knowledge of p and q.*

For efficiency reasons, it is often desirable to choose e to be a small integer, such as $e = 3$, or any other special form that makes encryption more efficient. Thus, the RSA assumption is often stated in a stronger form.

Definition 3.1.4. *Let $n = pq$ for distinct odd primes p and q. Given $c \in \mathbb{Z}_n$, the Strong RSA Problem is the problem of finding $m, e > 1$ such that $c = m^e \mod n$, without knowledge of p and q.*

An overview of the RSA and Strong RSA problems has been given by Rivest and Kaliski [87].

A slightly different form of the Strong RSA Problem is also used in some cryptosystems. Let λ be the Carmichael Function, such that $\lambda(n) = m$, the smallest positive integer such that $a^m \equiv 1 \mod n$ for all a relatively prime to n. The Carmichael function represents the exponent of the group $\mathbb{Z}_n^*$, and can be calculated by

$$\lambda(n) = \begin{cases} \phi(n) & \text{for } n = p^\alpha, \text{ with } p = 2 \text{ and } \alpha \leq 2, \text{ or } p \geq 3 \\ \frac{1}{2}\phi(n) & \text{for } n = 2^\alpha \text{ and } \alpha \geq 3 \\ \operatorname{lcm}(\lambda({p_i}^{\alpha_i})) & \text{for } n = \prod_i {p_i}^{\alpha_i} \end{cases}$$

The problem of computing e'th roots when $\gcd(e, \lambda(n^2)) = 1$ and $n = pq$ is called the *Computational Small e'th Roots Problem*, and was posed by Catalano, Gennaro, Howgrave-Graham, and Nguyen [25].

Definition 3.1.5. *Let $n = pq$ for distinct odd primes p and q. Given $e \in \mathbb{Z}_{n^2}^*$ such that $\gcd(e, \lambda(n^2)) = 1$, and an integer y, the Computational Small e'th Roots Problem is the problem of computing x such that $x^e = y \mod n^2$. The Decisional Small e'th Roots Problem is the problem of deciding whether or not a given x is an e'th root modulo n^2.*

Each of the e'th root problems can be parameterized by ϵ, the bit-length of n.

3.1.3 Quadratic Residuosity Problem

The problem of deciding whether or not an integer is a quadratic residue modulo n forms the basis of security for the Goldwasser-Micali cryptosystem, presented in Section 3.2.2. Further variations of residue problems also form the basis for most modern homomorphic cryptosystems.

Definition 3.1.6. *An integer a is said to be a quadratic residue modulo n if there exists $0 < x < n$ such that*

$$x^2 \equiv a \mod n.$$

Otherwise, a is said to be a non-quadratic residue modulo n.

If n is an odd prime, then determining whether or not an integer a is a quadratic residue modulo p is equivalent to calculating the Legendre symbol

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \equiv 0 \mod p \\ 1 & \text{if } a \not\equiv 0 \text{ and there exists } x \in \mathbb{Z} \text{ such that } a \equiv x^2 \mod p \\ -1 & \text{if no such } x \text{ exists} \end{cases}$$

which can be efficiently calculated by the formula

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \mod p.$$

For an odd composite number n, the Jacobi symbol can also be calculated

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}$$

where $n = {p_1}^{e_1}{p_2}^{e_2} \cdots {p_k}^{e_k}$ is the prime factorization of n. Unlike the Legendre symbol, $\left(\frac{a}{n}\right) = 1$ does not imply a is a quadratic residue modulo n.

Definition 3.1.7. *Let the set $\mathcal{QR}_n$ denote the set of quadratic residues modulo n. Given an odd composite integer $n > 3$ and an integer a such that $\left(\frac{a}{n}\right) = 1$, the problem of determining if $a \in \mathcal{QR}_n$ is called the Quadratic Residue Problem.*

If $n = pq$ for distinct odd primes p and q, then $a \in \mathcal{QR}_n$ if and only if $\left(\frac{a}{p}\right) = 1$ and $\left(\frac{a}{q}\right) = 1$. Thus, knowledge of the factorization of n is sufficient to solve the quadratic residuosity problem. It is currently believed, although unproved, that solving the quadratic residuosity problem is equivalent to factoring. Assuming the truth of this conjecture, the security parameter ϵ for the quadratic residuosity problem is the same as the security parameter for the factoring problem.

3.1.4 Higher Order Residues and Residue Classes

The difficulty of deciding whether or not an element is a quadratic residue, i.e., $z = x^2 \mod N$ for some x, remains difficult for higher powers as well.

Definition 3.1.8. *An integer z is said to be an r'th residue modulo N if there exists $y \in \mathbb{Z}_N^*$ such that*

$$z \equiv y^r \mod N$$

and y is said to be an r'th root of z.

The set of r'th residues forms a subgroup of $\mathbb{Z}_N^*$, and each r'th residue has the same number of r'th roots. Cohen and Fischer [35] have called the problem of deciding r'th residues the *Weak r'th Residue Problem.*

Definition 3.1.9. *Given a random $z \in \mathbb{Z}_N$, the problem of determining whether or not z is an r'th residue modulo N is called the Weak r'th Residue Problem.*

This problem is a natural generalization of the quadratic residuosity problem for higher order residues, and is assumed to be difficult when the factoring problem is hard with respect to N. Paillier [79] has considered the Weak r'th Residue Problem in the special case when $r = n$ and $N = n^2$ for $n = pq$ where p and q are distinct odd primes.

Definition 3.1.10. *Given a random $z \in \mathbb{Z}_{n^2}^*$, the problem of determining whether or not z is an n'th residue modulo n^2 is called the Composite Residuosity Problem, or CR[n].*

For fixed values of r and $N = n^2$, given an r'th residue $z = y^r \in \mathbb{Z}_N^*$, the set of integers of the form $g^x y^n$ are of particular interest.

Definition 3.1.11. *For fixed values of r and $N = n^2$, if $w \equiv g^x z \mod N$ for some r'th residue $z \equiv y^r \mod N$, then w is said to belong to the r'th residue class x with respect to g. The set of all $w \in \mathbb{Z}_N^*$ belonging to the r'th residue class x with respect to g is denoted $[w]_g$. That is,*

$$[w]_g = \{x \mid w \equiv g^x y^n \mod N, y \in \mathbb{Z}_n^*\}.$$

It can be shown that the different r'th residue classes form a partition of $\mathbb{Z}_N^*$; that is, every $w \in \mathbb{Z}_N^*$ belongs to at least one r'th residue class, and, $[w_1]_g \cap [w_2]_g \neq \emptyset \Rightarrow [w_1]_g = [w_2]_g$ for $w_1, w_2 \in \mathbb{Z}_N^*$. The problem of determining which r'th residue class w belongs is to conjectured to be a difficult problem.

For the remainder of this section, set $n = pq$ for distinct odd primes p and q, fix $r = n$ and $N = n^2$, and let $\lambda = \operatorname{lcm}(p-1, q-1)$ (see the Carmichael Function in Section 3.1.2). Paillier [79] has shown that if the order of g is a non-zero multiple of n, then $\mathcal{E}_g : \mathbb{Z}_n \times \mathbb{Z}_n^* \mapsto \mathbb{Z}_N^*$ defined by $\mathcal{E}_g(x, y) = g^x y^n \mod N$ is bijective. Additionally, if the order of g is αn for $\alpha \in \{1, \ldots, \lambda\}$, then g uniquely determines the residue class $x = [w]_g$ for a given $w \in \mathbb{Z}_N^*$. Thus, given w and g, there exists unique x, y such that $\mathcal{E}_g(x, y) = w$. The function $w \mapsto [w]_g$, for a specified g, is called the class function.

Definition 3.1.12. *Given $n = pq$ for distinct odd primes p and q, and $g, w \in \mathbb{Z}^*_{n^2}$, the problem of computing the class function $w \mapsto [w]_g$ is called the n'th Residuosity Class Problem in base g, or Class[n, g].*

It turns out that Class[n, g] is random self-reducible; that is, an instance of the problem for w can be transformed into an instance for some uniformly random $w' \in \mathbb{Z}^*_{n^2}$ in base g', such that the solution for w' also yields the solution for w. This implies that solving a specific instance of the problem is just as hard as solving a random instance of the problem. Thus, the difficulty of Class[n, g] depends only on n, allowing the problem to be restated as the *Composite Residuosity Class Problem*, or Class[n], as the computational problem of computing $[w]_g$ from w and g.

A solution to Class[n] implies a solution to CR[n], demonstrating that Class[n] is at least hard as CR[n], if not harder. This follows from the fact that w is an n'th residue if and only if $[w]_g = 0$ for a valid base g, i.e., $w = g^0 y^n = y^n$. The problem of deciding if $[w]_g = x$ for any valid x is called the *Decisional Composite Residuosity Class Problem*, or D-Class[n].

Definition 3.1.13. *Given $n = pq$ for distinct odd primes p and q, as well as $g, w \in \mathbb{Z}^*_{n^2}$ and $x \in \mathbb{Z}_n$, the problem of deciding if $[w]_g = x$ is called the Decisional Composite Residuosity Class Problem, or D-Class[n].*

A variant of Class[n] has also been proposed by Paillier, which limits the choice of w and y when computing the class function. This is motivated by forcing y to belong to $\langle g \rangle$, the subgroup generated by g. In this setting, a random $y \in \langle g \rangle$ can be represented by g^r for some random $r < |\langle g \rangle|$, and the computation of $\mathcal{E}_g(x, y) = g^x y^n$ becomes g^{x+nr}.

Definition 3.1.14. *Given $w \in \langle g \rangle$, the problem of computing $[w]_g$ from g and w is called the Partial Discrete Logarithm Problem (PDL[n, g]).*

The decisional version of PDL[n, g] is of interest as well.

Definition 3.1.15. *Given $w \in \langle g \rangle$ and $x \in \mathbb{Z}_n$, the problem of deciding whether or not $[w]_g = x$ is called the Decisional Partial Discrete Logarithm Problem (D-PDL[n, g]).*

As the names imply, these problems are related to the discrete logarithm problem, to be defined in Section 3.1.6.

Like Class[n], PDL[n, g] is also random self-reducible, although in a slightly different manner than Class[n].

Theorem 3.1.16. *Let $n = pq$ for distinct odd primes p and q, let $\lambda = lcm(p - 1, q - 1)$, and let H be the cyclic subgroup of $\mathbb{Z}^*_{n^2}$ of order λn. Then PDL[n, g] is random self-reducible over the cyclic subgroups of H.*

Proof. Given g such that $\langle g \rangle \leq H$ and $w \in \langle g \rangle$, it must be shown that a solution for PDL$[n, g]$ for a uniformly random $w' \in \langle g \rangle$ can be used to solve PDL$[n, g]$ for any given w. Choose α and β uniformly at random from $\mathbb{Z}_n$, then set $y = g^\beta$ and $w' = wg^\alpha y^n = wg^{\alpha+\beta n} \mod n^2$. Then w' is a uniformly random element of $\langle g \rangle$. If $w = g^{x+rn} \mod n^2$, then $w' = g^{x+rn}g^{\alpha+\beta n} = g^{(x+\alpha)+(r+\beta)n} \mod n^2$ and $[w']_g = x + \alpha \mod n$. If the solution to $[w']_g$ is known, then

$$[w]_g = [w']_g - \alpha = x$$

is easy to compute. Hence, PDL$[n, g]$ is random self-reducible over $\langle g \rangle$. $\square$

Because PDL$[n, g]$ is only random self-reducible over $\langle g \rangle$, the computational problem is parameterized by both n and g, unlike Class$[n]$, which relies only on n.

Each of the residuosity problems presented thus far are easy to solve if the factorization of n is known. It is currently unknown if the problems are polynomially equivalent, although it is generally not believed to be the case [79]. Because factoring n is still the best known approach to solving residuosity problems, the difficulty of the problem is usually parameterized by the bit-length of n.

3.1.5 The p-Subgroup Problem

Let p, q be distinct primes such that $p \not| q - 1$, and consider the group $\mathbb{Z}^*_{p^2q}$. The order of $\mathbb{Z}^*_{p^2q}$ is $\phi(p^2q) = p(p-1)(q-1)$. By Sylow's First Theorem [54], $\mathbb{Z}^*_{p^2q}$ has exactly one subgroup of order p, which is the Sylow p-subgroup. Deciding if an element $a \neq 1 \in \mathbb{Z}^*_{p^2q}$ belongs to the unique subgroup of order p can be verified by checking if $a^p \equiv 1 \mod p^2q$; i.e., the order of a in $\mathbb{Z}^*_{p^2q}$ is a divisor of p. Because p is prime, this can only occur if $|a| = 1$, which, by assumption, cannot occur, or $|a| = p$.

Verifying if an element belongs to the subgroup of order p is similar to the residue problems presented in the previous section, and it is conjectured to be difficult when then factorization of p^2q is unknown.

Definition 3.1.17. *Let $n = p^2q$ for distinct odd primes p and q such that $p \not| q - 1$. Then the problem of deciding whether or not a random element $a \neq 1 \in \mathbb{Z}^*_n$ is in the unique Sylow p-subgroup when the factorization of n is unknown is called the p-subgroup Problem.*

As with the residuosity problems, the best known approach to solving the p-subgroup problem is through factoring $n = p^2q$. Thus, the p-subgroup problem can be parameterized by ϵ, the bit-length of n.

3.1.6 Discrete Logarithms and the Diffie-Hellman Problem

The *discrete logarithm* is the group equivalent of the logarithm function for real numbers.

Definition 3.1.18. *Given a generator $\alpha \in \mathbb{G}$ of a cyclic group of order n, and given some $\beta = \alpha^x \in \mathbb{G}$, the discrete logarithm of β in base α is the unique value x* mod n. *Given β, the problem of finding x is called the* Discrete Logarithm Problem (DLP)*.*

The difficulty of solving DLP depends on the representation of the group considered. The most general setting is the *generic group model*, in which no assumptions are made about the underlying structure of the group. Shoup has shown that the time needed to solve DLP in the generic group model is at least $\Omega(\sqrt{n})$, where n is the size of the group [94]. Solutions to DLP that have expected running time $O(\sqrt{n})$, exist, showing this bound as tight. Such approaches include Pollard's rho algorithm, Pollard's lambda algorithm, and the baby-step giant-step algorithm. A survey of such algorithms can be found in [100].

A different approach, called the index-calculus method, does utilize the underlying group representation. If group elements can be efficiently represented using a factor base of small factors, such as integers whose prime factorizations contain only small prime powers, then the properties of the logarithm function can be used to reconstruct the logarithm of g from the logarithm of each of g's factors. A comprehensive survey of discrete logarithms and their uses in cryptography, including the index calculus method, is given in [76]. In general, the index-calculus method is more efficient than the generic square-root methods, so for cryptographic purposes groups are usually chosen such that the index calculus method does not work.

As seen earlier, many definitions of security for cryptographic systems rely on the adversary's inability to distinguish between ciphertexts. When building cryptosystems that rely the difficulty of solving DLP, this often translates to the adversary being unable to distinguish between the discrete logarithms of two group elements. The *Computational Diffie-Hellman Problem (CDH)* and *Decisional Diffie-Hellman Problem (DDH)* attempt to capture the difficulty of problems related to DLP as they naturally arise in cryptography.

Definition 3.1.19. *Given a generator $\alpha \in \mathbb{G}$, a cyclic group of order q, and two group elements α^a and α^b, the Computational Diffie-Hellman Problem (CDH) is the problem of calculating α^{ab} from α^a and α^b without knowledge of a and b.*

Clearly, a solution to DLP can be polynomially reduced to a solution to CDH, although it is currently unknown if a reduction in the other direction exists.

The decisional version of CDH requires the adversary to distinguish between α^{ab} and α^c for some random integer c, a situation that arises naturally from the IND-CPA game.

Definition 3.1.20. *Given a generator $\alpha \in \mathbb{G}$, a cyclic group of order q, and two group elements α^a and α^b, the Decisional Diffie-Hellman Problem (DDH) is the problem of distinguishing between tuples of the form $(\alpha^a, \alpha^b, \alpha^{ab})$ and $(\alpha^a, \alpha^b, \alpha^c)$ for some random integer c, without knowledge of a and b.*

As with CDH, a solution to DLP can be polynomially reduced to a solution to DDH; however, unlike CDH, there exist groups for which DDH is tractable, but DLP is not. One such example is an elliptic curve group that supports a bilinear pairing (described in the next section). Given a pairing e and elements α^a, α^b, and α^c, then one can compute $e(\alpha^a, \alpha^b) = e(\alpha, \alpha)^{ab}$ and $e(\alpha, \alpha^c) = e(\alpha, \alpha)^c$. If the two values are equal then $ab = c$, otherwise $ab \neq c$.

Assuming that groups are chosen for which the index-calculus method does not work, each of the discrete logarithm problems can be parameterized by ϵ, the bit-length of the size of the group. Thus, recalling the running time of DLP algorithms, ϵ must be chosen so that solving a problem of order $O(\sqrt{2^\epsilon})$ is intractable.

3.1.7 Bilinear Groups and Elliptic Curves

Elliptic curves play a very important role in modern cryptography. The presentation of elliptic curves given here is extremely limited, and is intended only to establish enough background information to present the Boneh-Goh-Nissim cryptosystem in Section 3.2.12 and the Elliptic Curve Paillier cryptosystem in Section 3.2.10. A more detailed treatment of the subject can be found in [10].

Many mathematical functions are linear, that is, the relation $f(ax) = af(x)$ holds for any x in the domain of f, and for any constant a. The concept of a linear function can be extended to higher dimensions, with the two-dimensional, or bilinear case, being of particular interest.

Definition 3.1.21. *Let $\mathbb{G}$ and $\mathbb{G}_1$ be two cyclic groups of order n with g a generator of $\mathbb{G}$. A map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is said to be bilinear if $e(g, g)$ is a generator of $\mathbb{G}_1$, and*

$$e(u^a, v^b) = e(u, v)^{ab}$$

for all $u, v \in \mathbb{G}$ and all $a, b \in \mathbb{Z}$.

In cryptography, bilinear functions, are usually generated from elliptic curve groups.

Definition 3.1.22. *An elliptic curve over the a field of characteristic not 2 or 3 is a plane algebraic curve defined by an equation of the form*

$$y^2 = x^3 + ax + b.$$

The set of points on an elliptic curve over a field whose characteristic is not 2 or 3, together with a special “point an infinity”, forms an abelian group. Boneh, Goh, and Nissim [14] describe the following method for constructing a bilinear pairing over such a group of order n, where n is square free and not divisible by 3:

1. Find the smallest integer l such that $p = ln - 1$ is prime and $p = 2 \mod 3$.
2. The curve $y^2 = x^3 + 1$ defined over $\mathbb{F}_p$ has $p + 1 = ln$ points in $\mathbb{F}_p$, and thus has a subgroup of order n, denoted by $\mathbb{G}$.
3. If $\mathbb{G}_1$ is the subgroup of order n of $\mathbb{F}_{p^2}$, then the modified Weil pairing [73] on the curve gives a bilinear map from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_1$.

The Weil pairing is a method of constructing a bilinear pairing on a special subgroup of an elliptic curve group over a field. Additional details can be found in [73].

When presenting an elliptic curve Paillier cryptosystem, Galbraith [53] uses a more general representation of elliptic curves, as the set of points $(x : y : z)$ such that $x, y, z \in R$, a commutative ring with unity, and

$$y^2 z = x^3 axz^2 + bz^3$$

where $6(4a^3 + 27b^2) \in R$ is invertible. Setting $z = 1$, this is the same as the formula in Definition 3.1.22.

3.2 A Survey of Homomorphic Cryptosystems

In this section a survey of known homomorphic cryptosystems is given, along with problems they rely on for security. When a proof of security for the cryptosystem has been provided by the authors, an outline of their proof is given. In most situations where proofs have been omitted or simplified by the authors, a proof is supplied.

The presentation of each cryptosystem differs slightly from the notation in Definition 1.3.1. For each cryptosystem presented, the plaintext space, ciphertext space, key space, and set of randomizers will be defined, and the cryptosystem will be presented with respect to three algorithms: *Gen*, *Enc*, and *Dec*. Given a security parameter, *Gen* creates an instance of the cryptosystem such that the difficulty of the underlying computational problem the cryptosystem relies on is based on a specified security parameter. *Enc* and *Dec* are algorithms that perform the encryption and decryption of messages given the public and private keys respectively.

3.2.1 The RSA Cryptosystem

The RSA cryptosystem, presented earlier as Cryptosystem 1.3.3, relies on the difficulty of calculating e'th roots modulo n, where $n = pq$ for distinct odd primes p and q, when the factorization of n is unknown. Although factoring n is sufficient to break RSA, it is unknown if breaking RSA is equivalent to factoring n. Boneh and Venkatesan [16] have given evidence suggesting that the problems are not equivalent in general. More specifically, they show that it cannot be proved that solving the RSA problem is as hard as factoring using a straight line program when the public exponent is small, unless factoring is easy. Later, Brown [18] demonstrated a result in the opposite direction; namely, it was shown that there is no efficient algorithm that takes a small public RSA exponent and outputs a straight line program that solves the RSA Problem, unless factoring is easy.

The RSA and Strong RSA problems presented in Section 3.1.2, have been defined to capture the precise problem of inverting the encryption function without knowledge of p and q.

Homomorphic Properties

The RSA cryptosystem provides the basic homomorphic operation of multiplication modulo n. Given two ciphertexts $c_1 = {m_1}^e \mod n$ and $c_2 = {m_2}^e \mod n$, then

$$\begin{aligned} c_1 c_2 \mod n &= {m_1}^e {m_2}^e \mod n \\ &= (m_1 m_2)^e \mod n \end{aligned}$$

is an encryption of $m_1 m_2$.

Unfortunately, the RSA cryptosystem is not IND-CPA secure unless messages are randomly padded. If m_1 and m_2 are randomly padded as ${m_1}'$ and ${m_2}'$, then recovering $m_1 m_2$ from ${m_1}'{m_2}'$ is impossible. For this reason, the RSA cryptosystem is not often used for its homomorphic properties.

3.2.2 The Goldwasser-Micali Cryptosystem

The Goldwasser-Micali (GM) cryptosystem [62] was the first probabilistic cryptosystem proposed, with its security relying on the difficulty of the quadratic residuosity problem. The GM cryptosystem encrypts only a single bit of information at a time, and does so by mapping a value of 0 to a random quadratic residue, and a value of 1 to a random non-quadratic residue with Jacobi symbol -1. Operations are performed modulo $n = pq$, where p and q are two distinct primes, with the factorization of n kept secret. Decrypting a ciphertext is equivalent to determining if a random integer is a quadratic residue modulo n, which, for an appropriate choice of n, is intractable without knowledge of p or q, as discussed in Section 3.1.3.

Cryptosystem 3.2.1. *The Goldwasser-Micali (GM) Cryptosystem*

Let $n = pq$, where p and q are distinct odd primes, and let $m \notin \mathcal{QR}_n$ with $\left(\frac{m}{n}\right) = -1$. Let $\mathcal{P} = \{0, 1\}$, $\mathcal{C} = \mathcal{R} = \mathbb{Z}_n^*$, and define $\mathcal{K} = \{(n, p, q, m)\}$ where n, p, q, m are as described above.

- *Gen: Given security parameter ϵ, Gen(ϵ) returns two distinct $\frac{\epsilon}{2}$-bit primes p and q, the value $n = pq$, as well as $m \notin \mathcal{QR}_n$. The tuple (n, m) is the public key, and the value p is the private key.*

- *Enc: Given a public key $pk = (n, m)$ and a message $x \in \mathcal{P}$, $Enc(pk, m)$ chooses a random value $r \in \mathcal{R}$ and returns the ciphertext*

$$c = m^x r^2 \mod n.$$

- *Dec: Given a private key $sk = p$ and a ciphertext $c \in \mathcal{C}$, $Dec(sk, c)$ returns the following message*

$$x = \begin{cases} 0 & \textit{if } y \in \mathcal{QR}_n \\ 1 & \textit{if } y \notin \mathcal{QR}_n. \end{cases}$$

A simplification of the system is possible by choosing p and q to be Blum integers. An integer $n = pq$ is a Blum integer if p and q are distinct primes with $p \equiv 3 \mod 4$ and $q \equiv 3 \mod 4$. When this is the case, -1 will always be a non-quadratic residue modulo n, and $(frac{-1}p) = \left(\frac{-1}{q}\right) = -1$, allowing the fixed the value of $m = -1$ to be used. This simplifies both the public key and the encryption process.

Theorem 3.2.2. *If the quadratic residuosity problem is hard with respect to n, then the GM cryptosystem is IND-CPA secure.*

The proof of this theorem follows from the fact that decrypting a ciphertext $c \in \mathbb{Z}_n^*$ is equivalent to determining if c is a quadratic residue modulo n. A complete proof can be found in [64].

A major downfall of the GM cryptosystem is the large size of ciphertexts relative to the single bit plaintexts. To remain secure, n must be at least 1024-bit, if not larger. Thus, messages encrypted under the GM cryptosystem grow by a factor of over 1000. Depending on the application, this may rule out the GM cryptosystem as a viable choice.

Homomorphic Properties

In addition to being a probabilistic cryptosystem, the GM cryptosystem also provides the homomorphic operation of XOR, or addition modulo 2. Consider two ciphertexts encrypted under the simplified GM cryptosystem, $c_1 = -1^{x_1}{r_1}^2$ and $c_2 = -1^{x_2}{r_2}^2$. Then

$$\begin{aligned} c_1 c_2 &= (-1^{x_1}{r_1}^2)(-1^{x_2}{r_2}^2) \\ &= -1^{(x_1+x_2)}(r_1 r_2)^2 \\ &= -1^{(x_1+x_2 \mod 2)}(r_1 r_2)^2 \end{aligned}$$

is an encryption of $x_1 + x_2 \mod 2$.

The GM cryptosystem also allows for a ciphertext to be re-randomized without knowledge of the plaintext. Given $c = -1^x {r_1}^2$, choose a random integer $r_2 \in \mathbb{Z}_n^*$

uniformly. Then the integer $r_3 = r_1 r_2 \mod n$ is a uniform random integer in $\mathbb{Z}_n^*$ and

$$\begin{aligned} c{r_2}^2 &= -1^x {r_1}^2 {r_2}^2 \\ &= -1^x {r_3}^2 \end{aligned}$$

is a random encryption of x.

3.2.3 The ElGamal Cryptosystem

The ElGamal cryptosystem [55] is a public-key cryptosystem based on the problem of solving discrete logarithms, making use of difficulty of solving both the CDH and DDH problems, as presented in Section 3.1.6.

Cryptosystem 3.2.3. *The ElGamal Cryptosystem*

Let $g \in \mathbb{G}$ be a generator of a cyclic group of order q such that the CDH and DDH problems are hard in $\mathbb{G}$. Let $\mathcal{P} = \mathbb{G}$, $\mathcal{C} = \mathbb{G} \times \mathbb{G}$, $\mathcal{R} = \mathbb{Z}_q$, and $\mathcal{K} = \{(q, g, x, h) : h \equiv g^x \mod q\}$, where q, g are as above.

- *Gen: Given security parameter ϵ, Gen(ϵ) returns a generator g of a cyclic group $\mathbb{G}$ of order q, where q has bit-length ϵ, as well as $h = g^x$ where x is a random integer in $\mathbb{Z}_q$. The tuple $(\mathbb{G}, q, g, h)$ is the public key, and the tuple $(\mathbb{G}, q, g, x)$ is the private key.*

- *Enc: Given a public key $pk = (\mathbb{G}, q, g, h)$ and a message $m \in \mathcal{P}$, Enc(pk, m) chooses a random $y \in \mathcal{R}$ and returns the ciphertext (c_1, c_2) where*

$$c_1 = g^y \mod q$$

and

$$c_2 = m \cdot h^y \mod q.$$

- *Dec: Given a private key $sk = (\mathbb{G}, q, g, h)$ and a ciphertext $c = (c_1, c_2)$, Dec(sk, c) returns the message*

$$\begin{aligned} \frac{c_2}{{c_1}^x} \mod q &= \frac{m \cdot h^y}{g^{xy}} \mod q \\ &= \frac{m \cdot g^{xy}}{g^{xy}} \mod q \\ &= m. \end{aligned}$$

The security of the ElGamal cryptosystem follows directly from the hardness of CDH and DDH when $\mathbb{G}$ is chosen appropriately.

Theorem 3.2.4. *If the CDH problem is hard in $\mathbb{G}$, then ElGamal encryption is not invertible by an adversary.*

Proof. If the CDH problem is hard in $\mathbb{G}$, then adversary cannot calculate g^{xy} from g^x and g^y. Assume the adversary can decrypt a ciphertext $(c_1, c_2) = (g^y, m \cdot g^{xy})$. Then, with knowledge of $h = g^x$ from the public key, and $c_1 = g^y$, the adversary can calculate $c_2 m^{-1} = g^{xy}$, contradicting the fact that CDH is hard. Thus, if CDH is hard, then encryption is a one-way function. □

Theorem 3.2.5. *If the DDH problem is hard in $\mathbb{G}$, then ElGamal ciphertexts are semantically secure.*

Proof. Let $\mathbb{G}$ be a group where the DDH problem is hard, and assume that the adversary has access to an oracle that can distinguish ElGamal ciphertexts in $\mathbb{G}$. Given an instance of the DDH problem, i.e., to distinguish (g^x, g^y, g^{xy}) and (g^x, g^y, g^r) for random $r \in \mathbb{G}$, the adversary can use the ElGamal oracle to solve DDH. Upon receiving the challenge (g^x, g^y, g^k) the adversary can construct the ciphertext $c = (g^y, g^k)$. If $g^k = g^{xy}$, then c is a valid encryption of 1. Otherwise, c is encryption of some random element of $\mathbb{G}$. The ElGamal oracle provides messages m_0 and m_1, allowing the adversary to compute $c_0 = (g^y, m_0 g^k)$ as an encryption of m_0 if $g^k = g^{xy}$, and some random element of $\mathbb{G}$ otherwise. The adversary sends c_0 to the oracle, who then responds with $b = 0$ if $g^k = g^{xy}$, or an error otherwise. If the oracle returns $b = 0$, then adversary concludes that the challenge is (g^x, g^y, g^{xy}). Otherwise, the adversary concludes that the challenge is (g^x, g^y, g^r). Hence, an oracle that breaks the semantic security of the ElGamal cryptosystem is polytime reducible to an algorithm that breaks DDH. Thus, if the DDH problem is hard, then the ElGamal cryptosystem is semantically secure. □

Homomorphic Properties

The ElGamal cryptosystem provides the homomorphic operation of multiplication of two encrypted messages, as well as multiplication by a known constant and exponentiation by a known constant. Given ciphertexts (c_1, c_2) and (d_1, d_2) that are encryptions of m_1 and m_2, using random values y_1 and y_2, respectively, then

$$\begin{aligned} (c_1 d_1, c_2 d_2) &= (g^{y_1} g^{y_2}, (m_1 \cdot h^{y_1})(m_2 \cdot h^{y_2})) \\ &= (g^{y_1 + y_2}, m_1 m_2 \cdot h^{y_1 + y_2}) \end{aligned}$$

is a valid encryption of $m_1 m_2$. Furthermore, given a constant k, then

$$(c_1, k c_2) = (g^{y_1}, k m_1 \cdot h^{y_1})$$

is a valid encryption of $k m_1$, and

$$({c_1}^k, {c_2}^k) = (g^{y_1 k}, {m_1}^k \cdot h^{y_1 k})$$

is a valid encryption of ${m_1}^k$.

ElGamal ciphertexts can also be re-randomized without knowledge of the plaintext. Given a ciphertext (c_1, c_2), as before, choose a random integer $r \in \mathbb{Z}_q$ and calculate

$$(c_1 g^r, c_2 h^r) = (g^{y_1 + r}, m \cdot h^{y_1 + r}),$$

which is a randomized valid encryption of m.

Variants

The ElGamal cryptosystem has been extended to provide IND-CCA2 security, as the Cramer-Shoup cryptosystem [38]; however, this comes at the cost of sacrificing malleability through homomorphic operations. Fujisaki and Okamoto [51] have shown a method for converting an IND-CPA secure cryptosystem into an IND-CCA2 cryptosystem, with ElGamal being one of the examples provided. Chevallier-Mames, Paillier, and Pointcheval [29] have investigated the problem of efficiently encoding messages as group elements in groups where the ElGamal cryptosystem is secure. They propose a variant which allows messages to be chosen directly from $\mathbb{Z}_q$, but at the cost of sacrificing some of the homomorphic properties of the system. In the modified cryptosystem, homomorphic operations are limited to multiplication or addition by a constant, and ciphertext re-randomization is not possible. Castagnos and Chevallier-Mames [22] have further studied the problem of encoding messages and have proposed a variant which borrows ideas from the Naccache-Stern cryptosystem, presented later in Section 3.2.5, which retains the homomorphic properties of the original cryptosystem as long as a simple assumption is satisfied.

Cramer, Gennaro and, and Schoenmakers [37] have given a threshold construction of the ElGamal cryptosystem, which is presented in Section 4.5. They have also presented a new variant of ElGamal which is additively homomorphic instead of multiplicatively homomorphic. They choose q such that $q|p-1$ for a prime p and replace the encryption function so that

$$(c_1, c_2) = (g^r \mod p, h^{r+m} \mod p)$$

for a random $r \in \mathbb{Z}_q$. Decryption then becomes

$$\begin{aligned} \log_h \frac{c_2}{c_1^x} &= \log_h \frac{h^{r+m}}{g^r} \\ &= \log_h \frac{g^{rx+xm}}{g^{rx}} \\ &= \log_h h^m. \end{aligned}$$

Thus, decryption requires the solution of a discrete logarithm, making the variant only suitable for situations when m is from a known small set. The Benaloh cryptosystem, presented in the next section, requires a similar computation, and techniques of recovering m are discussed within.

3.2.4 The Benaloh Cryptosystem

One of the negative aspects of the GM cryptosystem is the large message expansion. The plaintext space is a single bit, while the ciphertext operations are performed modulo n, where n is larger than 1024 bits. The fact that ciphertexts grow by a factor of $\log_2 n$ limits the practical use of the GM cryptosystem. This has motivated research into similar systems that retain probabilistic and homomorphic properties, but which have a more reasonable message expansion. Benaloh [9] has presented such a system, of which the GM cryptosystem is a special case.

The Benaloh cryptosystem allows a user to select a block length r, such that messages are chosen from $\mathbb{Z}_r$. The decreased message expansion comes at the cost of an increase in decryption complexity, which is increased to $O(\sqrt{r})$. In the case of the GM cryptosystem, $r = 2$ and decryption requires a single computation. The security of the Benaloh cryptosystem is based on the difficulty of deciding r'th residues, much like the GM cryptosystem is based on the difficulty of deciding quadratic residues.

Cryptosystem 3.2.6. *The Benaloh Cryptosystem*

Let r be an integer representing the desired block length of the cryptosystem, i.e., $\mathcal{P} = \mathbb{Z}_r$. Let $n = pq$ where p is a prime such that $r \,|\, p-1$, and q is a prime such that $\gcd(q-1, r) = 1$, and choose $y \in \mathbb{Z}_n^*$ such that $y^{(p-1)(q-1)/r} \not\equiv 1 \mod n$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_n^*$ and the keyspace is $\mathcal{K} = \{(r, n, p, q, y)\}$ where r, n, p, q, y are defined as above. Elements of $\mathcal{R} = \mathbb{Z}_n^*$ are used to randomize ciphertexts.

- *Gen: Given a security parameter ϵ and a block length r, Gen(ϵ, r) chooses an ϵ-bit integer $n = pq$ such that p is a prime with $r \,|\, p-1$, and q is a prime with $\gcd(q-1, r) = 1$, as well as $y \in \mathbb{Z}_n^*$ such that $y^{(p-1)(q-1)/r} \not\equiv 1 \mod n$. The public key is (y, n, r) and the private key is (p, q).*

- *Enc: Given a message $m \in \mathcal{P}$ and a public key $pk = (y, n)$, Enc(pk, m) chooses a random value $u \in \mathcal{R}$ and returns the ciphertext*

$$c = y^m u^r \mod n.$$

- *Dec: Given a private key $sk = (p, q)$ and a ciphertext c, Dec(sk, m) calculates*

$$m_i = (y^{-i}c)^{(p-1)(q-1)/r} \mod n$$

  for each $i \in \mathbb{Z}_r$ until $m_i = 1$, at which point it returns $m = i$.

The most complicated portion of the Benaloh cryptosystem is the decryption algorithm. Given $c = y^m u^r$, note that

$$\begin{aligned} c^{(p-1)(q-1)/r} &= (y^m u^r)^{(p-1)(q-1)/r} \\ &= y^{m(p-1)(q-1)/r}. \end{aligned}$$

Recalling that y was chosen so that $y^{(p-1)(q-1)/r} \not\equiv 1 \mod n$, then, since $m < r$, if $c^{(p-1)(q-1)/r} = y^{m(p-1)(q-1)/r} \equiv 1 \mod n$, it must be the case that $m = 0$. Similarly, for any $i \in \mathbb{Z}_r$,

$$\begin{aligned} m_i &= (y^{-i}c)^{(p-1)(q-1)/r} \mod n \\ &= y^{-i(p-1)(q-1)/r} y^{m(p-1)(q-1)/r} \mod n \\ &= y^{(m-i)(p-1)(q-1)/r} \mod n \end{aligned}$$

is equivalent to 1 modulo n if and only if $m - i = 0$; i.e., $m = i$.

The decryption process can be sped up by pre-computing the values

$$T_m = y^{m(p-1)(q-1)/r}$$

for each $m \in \mathbb{Z}_r$. Then, the value $T = c^{(p-1)(q-1)/r}$ be looked up in the table of pre-computed values. If $T = T_i$, then $m = i$. An alternative to a complete lookup table is to use the baby-step giant-step approach for solving discrete logarithms [100]. One can store a table of $O(\sqrt{r})$ pre-computed values to determine a good starting point, and then linearly search for the correct value to find the decryption of c using time and space $O(\sqrt{r})$.

The security of the Benaloh cryptosystem depends on the weak r'th residue problem, as presented in Definition 3.1.9. A formal proof of security for the cryptosystem is not given in [9], but, in the same manner that distinguishing quadratic residues forms the basis of security for the Goldwasser-Micali cryptosystem, as decrypting a GM ciphertext is exactly solving the quadratic residuosity problem, distinguishing messages in the Benaloh cryptosystem is the same as determining which value of m_i is an r'th residue.

Theorem 3.2.7. *The Benaloh cryptosystem is semantically secure if and only if the weak r'th residue problem is hard.*

Proof. (Outline) Assume the adversary selects $m_0 = 0$ and $m_1 = 1$ in step 3 of the semantic security game, and receives the challenge ciphertext $c = m_b$. Determining if $b = 0$ is exactly determining if c is an r'th residue. If the adversary selects $m_0 = k_0$ and $m_1 = k_1$, then determining if $b = 0$ is exactly determining if c^{-k_0} is an r'th residue. $\square$

Although the Benaloh cryptosystem is much more space efficient than the GM cryptosystem, the increased cost of decryption limits its uses. It is ideal for applications such as cryptographic voting, where many encryptions of ballots and many homomorphic operations for tallying are required (both of which are relatively efficient in the Benaloh cryptosystem), but only one decryption is required to determine the final tally. This is not a coincidence, as almost a decade before its general presentation in [9], the Benaloh cryptosystem was presented by Cohen (Benaloh) and Fischer [35], as well as in Benaloh's PhD thesis [8], in the context of verifiable secret-ballot elections, where the block size is a prime p selected to be a bound on the number of candidates on the ballot. The homomorphic properties of the cryptosystem were used to efficiently compute an encrypted tally of the votes.

Many homomorphic cryptosystems discovered after the Benaloh cryptosystem make use of special groups for which it is possible to efficiently solve certain discrete logarithms, so long as the private key is known. Such cryptosystems support decryption algorithms that are much more efficient than the Benaloh cryptosystem, allowing them to be used in a wider range of applications.

Homomorphic Properties

The Benaloh cryptosystem supports the homomorphic addition and subtraction of ciphertexts. Given two ciphertexts $c_1 = y^{m_1}{u_1}^r$ and $c_2 = y^{m_1}{u_1}^r$,

$$\begin{aligned} c_1c_2 \mod n &= (y^{m_1}{u_1}^r)(y^{m_2}{u_2}^r) \mod n \\ &= y^{m_1+m_2}(u_1u_2)^r \mod n \end{aligned}$$

is a valid decryption of $m_1 + m_2$, and

$$\begin{aligned} c_1{c_2}^{-1} \mod n &= (y^{m_1}{u_1}^r)(y^{m_2}{u_2}^r)^{-1} \mod n \\ &= (y^{m_1}{u_1}^r)(y^{-m_2}(u_2^{-1})^r) \mod n \\ &= y^{m_1-m_2}(u_1{u_2}^{-1})^r \mod n \end{aligned}$$

is a valid encryption of $m_1 - m_2$. Additionally, multiplication and exponentiation by a known constant k is also possible:

$$c_1y^k \mod n = y^{m+k}u^r \mod n$$

and

$$c_1{}^k \mod n = y^{mk}(u^k)^r \mod n.$$

Like the Goldwasser-Micali cryptosystem, the Benaloh system also supports the re-randomization of messages without knowledge of the plaintext. Given a ciphertext $c = y^m u^r$, choose a random value $u' \in \mathbb{Z}_n$ and calculate

$$cu'^r \mod n = y^m(uu')^r \mod n$$

as a random valid encryption of m.

3.2.5 The Naccache-Stern Cryptosystem

The Naccache-Stern cryptosystem [74] can be viewed as a generalization of the Benaloh cryptosystem. In the Benaloh cryptosystem, messages are bounded by a small prime p, whereas in the Naccache-Stern cryptosystem messages are bounded by the product of many small primes. An encrypted message is recovered by decrypting it modulo each of the small primes, and then reconstructing the message using Chinese remaindering. This allows the Naccache-Stern cryptosystem to achieve a smaller message expansion that the Benaloh cryptosystem.

Theorem 3.2.8. *The Chinese Remainder Theorem*

If $p_1, \dots, p_k$ are pairwise coprime integers, then given any integers $m_1, \dots, m_k$, the set of linear congruences $m \equiv m_i \mod p_i$ for $i = 1, \dots, k$ has a unique solution modulo $n = p_1 \cdots p_k$. Furthermore, the solution can be calculated by

$$m = \sum_{i=1}^{k} m_i N_i M_i \mod n$$

where $N_i = \frac{n}{p_i}$ and $M_i = {N_i}^{-1} \mod p_i$ for $i = 1, \dots, k$.

The basic Naccache-Stern cryptosystem is defined as follows:

Cryptosystem 3.2.9. *The Naccache-Stern Cryptosystem (Basic)*

Let $\{p_1, \ldots, p_k\}$ be a family of small distinct odd primes where k is even and set $u = \prod_{i=1}^{k/2} p_i$, $v = \prod_{i=k/2+1}^{k} p_i$, and $\sigma = uv$. Let a and b be primes such that $p = 2au + 1$ and $q = 2bv + 1$ are prime, and set $n = pq$. Let $g \in \mathbb{Z}_n^*$ such that the order of g is a large multiple of σ. Then $\mathcal{P} = \mathbb{Z}_\sigma$, $\mathcal{C} = \mathbb{Z}_n^*$, and $\mathcal{K} = \{(n, g, p, q)\}$ where n, g, p, q are defined as above.

- *Gen: Given a security parameter ϵ, Gen(ϵ) chooses an even k and generates distinct odd primes $p_1, \ldots, p_k$, sets $u = \prod_{i=1}^{k/2} p_i$, $v = \prod_{i=k/2+1}^{k} p_i$, and finds primes a, b such that $p = 2au + 1$ and $q = 2bv + 1$ are $\frac{\epsilon}{2}$-bit primes. Then $n = pq$, and g is selected at random such that the order of g is $\frac{\phi(n)}{4}$. The public key is (n, g) and the private key is (p, q).*

- *Enc: Given a public key pk and a message m, Enc(pk, m) calculates the ciphertext*
$$c = g^m \mod n.$$

- *Dec: Given a private key sk and a ciphertext c, Dec(sk, c) first calculates*
$$c_i = c^{\frac{\phi(n)}{p_i}} \mod n$$
and $m_i = j$, for the unique $0 \leq j < p_i$ such that
$$g^{\frac{j\phi(n)}{p_i}} \equiv j \mod n$$
for $i \in \{1, \ldots, k\}$. The message is then recovered through Chinese remaindering on each message $m_i \mod p_i$
$$m = \sum_{i=1}^{k} m_i N_i M_i \mod n$$
where $N_i = \frac{n}{p_i}$ and $M_i = N_i^{-1} \mod p_i$.

During decryption, the c_i's are calculated as

$$\begin{aligned} c_i &= g^{\frac{m\phi(n)}{p_i}} \mod n \\ &= g^{\frac{(m+m_i-m_i)\phi(n)}{p_i}} \mod n \\ &= g^{\frac{m_1\phi(n)}{p_i}} g^{\frac{m-m_i}{p_i}\phi(n)} \mod n \\ &= g^{\frac{m_1\phi(n)}{p_i}} \mod n \end{aligned}$$

where the term $g^{\frac{m-m_i}{p_i}\phi(n)}$ disappears due to the fact that the order of g divides $\frac{m-m_i}{p_i}\phi(n)$. The value of m_i is solved by a brute force search over $0 \leq j < p_i$, yielding the equation $m \equiv m_i \mod p_i$. Once each m_i is calculated, the message is recovered via Chinese remaindering.

It is possible to make the basic Naccache-Stern cryptosystem probabilistic with very few modifications.

Cryptosystem 3.2.10. *The Naccache-Stern Cryptosystem (Probabilistic)*

Let $n, p, q, \sigma, \mathcal{P}, \mathcal{C}$ be as in Cryptosystem 3.2.9. Then $\mathcal{R} = \mathbb{Z}_n$ and $\mathcal{K} = \{(n,\ g,\ p,\ q\ ,\sigma)\}$ where n, g, p, q, σ are defined as in Cryptosystem 3.2.9.

- *Gen: See Cryptosystem 3.2.9, and let $\sigma = uv$. The public key is (n, g, σ) and the private key is (p, q).*
- *Enc: Given a public key pk and a message $m \in \mathcal{P}$, Enc(pk, m) chooses a random $x \in \mathcal{R}$ and calculates the ciphertext*

$$c = g^m x^\sigma \mod n.$$

- *Dec: See Cryptosystem 3.2.9.*

Even though the random factor x is introduced during encryption, the decryption algorithm remains unchanged. Each c_i is calculated as

$$\begin{aligned} c_i &= g^{\frac{m\phi(n)}{p_i}} x^{\frac{\sigma\phi(n)}{p_i}} \mod n \\ &= g^{\frac{m\phi(n)}{p_i}} x^{\frac{\sigma}{p_i}\phi(n)} \mod n \\ &= g^{\frac{m\phi(n)}{p_i}} \end{aligned}$$

where the x vanishes due to the fact that $\frac{\sigma}{p_i}\phi(n)$ is a multiple of the order of $\mathbb{Z}_n^*$.

The security of the basic Naccache-Stern cryptosystem relies on both the difficulty of factoring n, and the difficulty of solving discrete logarithms. If an adversary is able to take the discrete logarithm in base g, then the message is trivially recovered. Similarly, if the factorization of n is known, then the adversary can perform the decryption algorithm to recover the message. Naccache and Stern note that although inverting the encryption function is not known to be equivalent to factoring, there are no other known attacks against the cryptosystem. Thus, the security of the Naccache-Stern cryptosystem is simply conjectured to hold, but has no formal proof.

The semantic security of the probabilistic version relies on the difficulty of deciding p_i'th residues for each of the p_i's, although σ must be chosen such that it reveals no information about the factorization of n. Recall that the value σ is part of the public key, and that $\phi(n) = (p-1)(q-1) = (2au)(2bv) = 4ab\prod_{i=1}^{k} p_i = 4ab\sigma$. Thus, $4ab = \frac{\phi(n)}{\sigma}$, and $\phi(n) = n-p-q+1$. Given that p and q should have the same bit-length, an adversary can approximate the bit-length of ab with high accuracy by approximating the bit-length of $n - p - q + 1$. Naccache and Stern provide a rather technical proof that parameters can be chosen appropriately such that the adversary cannot guess the value of ab, which relies on the fact that $n >> \sigma^4$. Thus, to remain secure, the ratio of the bit-lengths of σ and n must satisfy $\frac{|\sigma|}{|n|} < \frac{1}{4}$.

Theorem 3.2.11. *([74], Theorem 1) The Naccache-Stern cryptosystem is semantically secure if and only if the problem of deciding p_i'th residues for each p_i is hard.*

An outline of the proof of this theorem is provided as an appendix in [74], which relies on the hybrid technique as described in [60].

Recall that messages are chosen from $\mathbb{Z}_\sigma$, and are encrypted as elements of $\mathbb{Z}_n^*$. Thus, the message expansion is $\frac{|n|}{|\sigma|}$, where $|\sigma|$ and $|n|$ are the bit-lengths of σ and n. To remain secure, $\frac{|n|}{|\sigma|} > 4$ must be satisfied, providing a lower bound of 4 on the message expansion. The Okamoto-Uchiyama cryptosystem, presented Section 3.2.7, was discovered at the same time as the Naccache-Stern cryptosystem, and attracted more attention due to the fact that it easier to implement, has security that provably relies on the difficulty of the factoring problem, and also has a fixed message expansion rate of 3. For these reasons, the Naccache-Stern cryptosystem has not received as much attention as the Okamoto-Uchiyama cryptosystem and its successors.

Homomorphic Properties

The Naccache-Stern cryptosystem allows for the homomorphic addition and subtraction of ciphertexts, as well as multiplication of a ciphertext by a constant. Examples for the probabilistic variant are provided, but work equivalently for the basic cryptosystem.

Given two ciphertexts $c_1 = g^{m_1}{x_1}^\sigma \mod n$ and $c_2 = g^{m_2}{x_2}^\sigma \mod n$ as valid encryptions of m_1 and m_2 respectively, then

$$\begin{aligned} c_1c_2 \mod n &= g^{m_1}{x_1}^\sigma g^{m_2}{x_2}^\sigma \mod n \\ &= g^{m_1+m_2}(x_1x_2)^\sigma \mod n \end{aligned}$$

is a valid encryption of $m_1 + m_2$ and

$$c_1{}^k \mod n = g^{km_1}({x_1}^k)^\sigma \mod n$$

is a valid encryption of km_1. Subtraction is possible by taking $c_1{c_2}^{-1}$.

Implementation Details

Decryption of a ciphertext requires the computation of $c_i = c^{\frac{\phi(n)}{p_i}}$. In practice, it is more efficient to compute $c' = c^{4ab} \mod n$ and then calculate $c_i = c'^{\frac{\sigma}{p_i}} \mod n$ for each pre-computed value $\sigma {p_i}^{-1} \mod \phi(n)$. If modular exponentiation is performed through repeated squaring, then the intermediate values can also be stored in a look-up table after their first computation. Additionally, decryption can be performed twice modulo p and q, significantly reducing the complexity of each modular operation, and the message can be reconstructed modulo n through Chinese remaindering.

3.2.6 The Sander-Young-Yung Cryptosystem

Sander, Young, and Yung [90] have investigated methods for evaluating certain circuits involving OR and NOT gates in a two party setting, where one party has knowledge of a circuit computing some secret function f, and the other has a secret input x for which it would like to learn $f(x)$. This protocol is presented later in Chapter 5. Although their protocol could be used to calculate AND by DeMorgan's theorem, they also present a separate AND-homomorphic cryptosystem based on the Goldwasser-Micali cryptosystem, as described in Section 3.2.2.

The Goldwasser-Micali cryptosystem takes a message $m \in \{0, 1\}$, and encrypts it as a random quadratic residue if $m = 0$, and a random non-quadratic residue if $m = 1$. By taking the product of two ciphertexts, one can compute an encryption of the XOR of the two plaintext messages. The AND-homomorphic cryptosystem given by Sander, Young, and Yung works by adding an additional level of encoding on top of the Goldwasser-Micali cryptosystem. Given a positive non-zero integer l, consider a function *Encode* that maps an element of $\mathbb{Z}_2$ to a vector in $(\mathbb{Z}_2)^l$ such that *Encode*(0) is a random non-zero vector, and *Encode*(1) is the zero vector. The function *Decode* inverts *Encode* by mapping all non-zero vectors to 0, and the zero-vector to 1. Given two vectors, the AND operation is performed by computing the component-wise XOR of the two vectors. If both vectors are the zero vector, i.e., represent a value of 1, then the resulting vector v is also the zero vector, and *Decode*$(v) = 1$. If one or both of the vectors are not the zero vector, i.e., encode a value of 0, then the resulting vector v is a non-zero vector, except with the two vectors are equal (occurring with probability $\frac{1}{2^l}$), and *Decode*$(v) = 0$ with probability $1 - \frac{1}{2^l}$. Thus, given two bits b_0 and b_1, $Decode(Encode(b_0) \oplus Encode(b_1)) = b_0 \oplus b_1$ with probability $1 - \frac{1}{2^l}$.

Cryptosystem 3.2.12. *Sander-Young-Yung AND-Homomorphic Cryptosystem*

Let $n = pq$ for distinct odd primes p and q, and let l be a positive non-zero integer. Let Encode : $\mathbb{Z}_2 \to (\mathbb{Z}_2)^l$ be a function that maps 0 to a random non-zero vector, and 1 to the zero vector, and let Decode be its inverse. Then $\mathcal{P} = \mathbb{Z}_2$, $\mathcal{C} = (\mathbb{Z}_n)^l$, $\mathcal{R} = (\mathcal{QR}_n)^l$, where $\mathcal{QR}_n$ is the set of quadratic residues modulo n, and $K = \{(n, p, q, m, l) : m \in \mathcal{QR}_n\}$ where n, p, q and l are defined as above.

- *Gen: Given security parameter ϵ and a positive non-zero integer l, $Gen(\epsilon, l)$ creates an instance of the Goldwasser-Micali cryptosystem using security parameter ϵ, which returns (n, p, q, m). The public key is (n, m, l) and the private key is p.*

- *Enc: Given a message $m \in \mathcal{P}$ and a public key pk, $Enc(pk, m)$ uses the method $Encode(m)$ to generate a vector $v \in (\mathbb{Z}_2)^l$, and returns the ciphertext vector c, where each component of the vector is encrypted under the Goldwasser-Micali cryptosystem. That is,*

  $$c = E(pk, v) = (E(pk, v_1), E(pk, v_2), \ldots, E(k, v_l))$$

  where E is the encryption function from the Goldwasser-Micali cryptosystem.

- *Dec: Given a ciphertext c and a private key sk, $Dec(sk, c)$ calculates*

$$v = D(sk, c) = (D(sk, c_1), D(sk, c_2), \ldots, D(sk, c_l))$$

where D is the decryption function from the Goldwasser-Micali cryptosystem. The message is recovered by

$$m = Decode(v).$$

The non-inveritbility of the Sander-Young-Yung cryptosystem follows directly from the Goldwasser-Micali cryptosystem.

Theorem 3.2.13. *The Sander-Young-Yung encryption function is not invertible by the adversary if and only if the Goldwasser-Micali encryption function is not invertible by an adversary.*

Proof. (Outline) If an adversary can invert the GM encryption function, the adversary can invert each component of the ciphertext c to find the vector v, and calculate $Decode(v)$ to learn the message. Conversely, if an adversary can invert the SYY cryptosystem with for some value of l, then given a GM ciphertext c, the adversary can construct the encrypted vector $c' = (c, Enc(pk, 0), \ldots, Enc(pk, 0))$. The adversary then decrypts c' to learn m', thus learning the decryption of c as the compliment of m'. Hence, the SYY cryptosystem is not invertible by an adversary if and only if the GM cryptosystem is not invertible by an adversary. □

The semantic security of the Sander-Young-Yung cryptosystem also follows directly from the Goldwasser-Micali cryptosystem.

Theorem 3.2.14. *The Sander-Young-Yung cryptosystem is semantically secure if and only the Goldwasser-Micali cryptosystem is semantically secure.*

Proof. (Outline) If an adversary can break the semantic security of the GM cryptosystem, then the adversary can determine whether or not the vector c encrypted under the SYY cryptosystem contains an encryption of 1. If so, the adversary concludes that the vector represents an encryption of 0. Otherwise, the adversary concludes that the vector is a encryption of 1. Conversely, if an adversary can break the semantic security of the SYY cryptosystem, then given a GM ciphertext c, the adversary can construct the encrypted vector $c' = (c, Enc(pk, 0), \ldots, Enc(pk, 0))$. If the adversary determines that c' is an encryption of 0, then the adversary can conclude that c is an encryption of 1. Otherwise, the adversary concludes that c is an encryption of 0. Hence, the SYY cryptosystem is semantically secure if and only if the GM cryptosystem is semantically secure. □

Although of theoretical interest as the first AND homomorphic cryptosystem, the Sander-Young-Yung cryptosystem amplifies the already serious problem of ciphertext expansion in the Goldwasser-Micali cryptosystem. To remain secure against modern factoring techniques, n must be at least 1024 bits in length. Thus, a ciphertext representing a single plaintext bit is of size $1024l$, where l is likely to be 32 or larger.

Homomorphic Properties

Recall that the homomorphic operation of the Goldwasser-Micali cryptosystem, performed by taking the product of two ciphertexts, is a binary XOR. GM ciphertexts can also be efficiently re-randomized by multiplying a ciphertext by a random encryption of 0. In a similar manner, ciphertexts in the Sander-Young-Yung cryptosystem can be efficiently re-randomized by taking the component-wise product with an encryption of 0 under the SYY cryptosystem.

Given two ciphertexts x and y as encryptions of m_1 and m_2 respectively, an encryption of $m_1 \cdot m_2$ can be calculated by first choosing two random non-singular matrices $A, B \in (\mathbb{Z}_2)^{l \times l}$, and then calculating $c = Ax + By$ by

$$c_i = \left(\prod_{\substack{j \\ a_{i,j}=1}} x_j \right) \left(\prod_{\substack{j \\ b_{i,j}=1}} b_{i,j} \right).$$

The resulting ciphertext c is then re-randomized using the method described earlier.

The homomorphic AND operation works by using the homomorphic XOR properties of the Goldwasser-Micali cryptosystem to do a component-wise homomorphic XOR on the two encrypted vectors. If both $m_1 = m_2 = 1$, then both c_1 and c_2 are encryptions of the zero vector, and a component-wise XOR yields the zero vector. Otherwise, a random non-zero vector is calculated, except in the case $c_1 = c_2$. In order to ensure this occurs with probability $\frac{1}{2^l}$, the two matrices A and B are used to randomize the encryptions of x and y. Finally, the new ciphertext is re-randomized so that it represents a random encryption of the result.

3.2.7 The Okamoto-Uchiyama Cryptosystem

The Okamoto-Uchiyama cryptosystem [77] is is a public-key cryptosystem based on the difficulty of the factoring problem. Unlike RSA, and most other cryptosystems based on factoring, the Okamoto-Uchiyama cryptosystem relies on the difficulty of factoring numbers of the form $n = p^2q$, rather than $n = pq$. Although the factorization of numbers of the form p^2q is less studied than numbers of the form pq, the time complexity of most factoring algorithms depends on the size of n, or the size of the factors of n, which imply that p and q can be selected such that $n = p^2q$ is large enough to resist known factoring techniques.

The Okamoto-Uchiyama cryptosystem improves on previous cryptosystems by working over groups of a special form, for which it is possible to calculate discrete logarithms in certain subgroups of $\mathbb{Z}_n^*$ when the factorization of n is known. For a prime p, consider the function $L(x) = \frac{x-1}{p}$ defined over the unique subgroup H of order p of $\mathbb{Z}_{p^2}^*$, which exists due to Sylow's First Theorem [54] and the fact that $|\mathbb{Z}_{p^2}^*| = p(p-1)$.

Lemma 3.2.15. *([77], Lemma 2) For $a, b \in H$,*

$$L(ab \mod p^2) = L(a) + L(b) \mod p,$$

and L is an isomorphism.

Proof. L is clearly bijective, and

$$\begin{aligned} L(ab) &= \frac{ab-1}{p} \mod p \\ &= \frac{(a-1)(b-1)+(a-1)+(b-1)}{p} \mod p \\ &= L(a)(b-1)+L(a)+L(b) \mod p \\ &= L(a)+L(b) \mod p \quad (\text{note: } b \equiv 1 \mod p). \end{aligned}$$

Hence, L is an isomorphism. □

The map L can be treated as a logarithm function with base p from the multiplicative group H to the additive group $\mathbb{Z}_p$. Thus, it is possible to use L to convert a logarithm in base p to a logarithm in base g, for some other group element g. This forms the basis for the Okamoto-Uchiyama cryptosystem.

Cryptosystem 3.2.16. *The Okamoto-Uchiyama Cryptosystem*

Let $n = p^2q$, where p and q are distinct k-bit primes, let g be a random element of $\mathbb{Z}_n^*$ such that g^{p-1} has order p in $\mathbb{Z}_{p^2}^*$, and set $h = g^n \mod n$. Then $\mathcal{P} = \mathbb{Z}_p$, $\mathcal{C} = \mathbb{Z}_n^*$, $\mathcal{R} = \mathbb{Z}_n$, and $\mathcal{K} = \{(n,p,q,g,h,k)\}$ where (n,p,q,g,h,k) are defined as above.

- *Gen: Given a security parameter ϵ, Gen(ϵ) sets $k = \frac{\epsilon}{3}$ and chooses $n = p^2q$, where p and q are distinct k-bit primes, then chooses a random element $g \in \mathbb{Z}_n^*$ such that $g^{p(p-1)} \equiv 1 \mod p^2$ and $g^{p-1} \not\equiv 1 \mod p^2$. Setting $h = g^n \mod n$, the public key is (n,g,h,k) and the private key is (p,q).*

- *Enc: Given a message $m \in \mathcal{P}$ and a public key $pk = (n,g,h,k)$, Enc(pk, m) chooses a random $r \in \mathcal{R}$ and returns the ciphertext*

$$c = g^m h^r \mod n.$$

- *Dec: Given a ciphertext c and a private key sk, Dec(sk, c) returns the message*

$$m = \frac{L(c^{p-1} \mod p^2)}{L(g^{p-1} \mod p^2)} \mod p$$

where

$$L(x) = \frac{x-1}{p}.$$

During decryption, the numerator simplifies to

$$\begin{aligned} L(c^{p-1} \mod p^2) &= L((g^m h^r)^{p-1} \mod p^2) \\ &= L((g^m g^{nr})^{p-1} \mod p^2) \\ &= L((g^{p-1})^m \; g^{p(p-1)(rpq)} \mod p^2) \\ &= L((g^{p-1})^m \mod p^2) \end{aligned}$$

making the decryption calculation

$$\frac{L((g^{p-1})^m \mod p^2)}{L(g^{p-1} \mod p^2)} \mod p.$$

Because the function L acts as a logarithm function, the decryption formula actually converts the logarithm of $(g^{p-1})^m$ in base g, to the same logarithm in base g^{p-1}, thus yielding the message m. This follows from Lemma 3.2.15.

It can be shown that an adversary that can decrypt a ciphertext c representing an encryption of m, can also factor n, demonstrating that the ability to decrypt a message is identical to factoring $n = p^2q$.

Theorem 3.2.17. *([77], Theorem 6) Inverting the Okamoto-Uchiyama encryption function without knowledge of (p, q) is intractable if and only if the factoring problem for $n = p^2q$ is intractable.*

Proof. (Outline) Clearly knowledge of (p, q) is sufficient to decrypt a ciphertext c. It remains to be seen that decrypting c allows the factorization of n. Let z be a random element of $\mathbb{Z}_n$, and set $c' = g^z$. With overwhelming probability, $z \notin \mathbb{Z}_p$. If the adversary can successfully decrypt c', yielding some correct message m, then $m \in \mathbb{Z}_p$ and $z \equiv m \mod p$. Because $z \notin \mathbb{Z}_p$ with overwhelming probability, $z \not\equiv m \mod n$ holds with overwhelming probability. Thus, the adversary can calculate $\gcd(z - m, n)$, yielding a non-trivial factor of n, which then allows n to be fully factored. Hence, decryption of a ciphertext is equivalent to factoring n. $\square$

The fact that inverting the encryption function reduces to factoring $n = p^2q$ implies a weakness of the cryptosystem with respect to an active adversary. The homomorphic properties of the Okamoto-Uchiyama cryptosystem prevent it from attaining security against adaptive adversaries; however, it is not the security of a single message, but the security of the entire system that is affected. Because inverting the encryption function is equivalent to factoring n, an adversary with access to a decryption oracle can factor n with non-negligible probability. As in the proof of Theorem 3.2.17, the adversary chooses a random $c' \in \mathbb{Z}_n$ submits a ciphertext $c' = g^z$ to the decryption oracle, which responds with m such that there exists h, r with $m = g^m h^r \mod n$. Because $z \notin \mathbb{Z}_p$ with high probability, it must be the case that $z \not\equiv m \mod n$ and $\gcd(z-m, n)$ will yield a proper factor of n. Thus, an active adversary can learn the private key and compromise the security of the entire cryptosystem. This type of attack is often referred to as a Rabin-style attack, due to the fact that it was first observed against the Rabin cryptosystem [85], the first cryptosystem discovered whose security is provably equivalent to factoring.

The semantic security of the system relies on the difficulty of the p-subgroup problem, as described in Section 3.1.5.

Theorem 3.2.18. *The Okamoto-Uchiyama cryptosystem is semantically secure if and only if the p-subgroup problem is intractable.*

Proof. (Outline) In the Okamoto-Uchiyama cryptosystem, the p-subgroup problem is equivalent to distinguishing between valid encryptions of 0 and 1. Let $c_0 = h^{r_0} \mod n$ be a valid encryption of 0, and $c_1 = gh^{r_1} \mod n$ be a valid encryption of 1, and recall that $h = g^n \mod n$. It can be determined if c_0 or c_1 have order p by calculating

$$\begin{aligned} {c_0}^{p-1} \mod p^2 &= (g^{nr_0})^{p-1} \mod p^2 \\ &= g^{(p-1)p^2qr_0} \mod p^2 \\ &= (g^{p(p-1)})^{pqr_0} \mod p^2 \\ &= 1 \end{aligned}$$

and

$$\begin{aligned} c_1^{p-1} \mod p^2 &= (g \cdot g^{nr_1})^{p-1} \mod p^2 \\ &= g^{p-1} \cdot g^{p^2qr_1+1} \mod p^2 \\ &= g^{p-1} \cdot (g^{(p-1)p})^{pqr_1} \mod p^2 \\ &= g^{p-1} \mod p^2. \end{aligned}$$

Thus, $|c_0| = n$ and $|c_1| = p$. Utilizing the fact that the p-subgroup problem is equivalent to distinguishing c_0 and c_1, it is possible to show that the ability to distinguish between c_0 and c_1 (i.e., break the p-subgroup assumption), allows an adversary to distinguish between encryptions of two arbitrary messages m_0 and m_1. Conversely, the ability to distinguish between encryptions of m_0 and m_1, allows an adversary to distinguish between c_0 and c_1, thus breaking the p-subgroup assumption. A complete proof is found in [77]. $\square$

One of the drawbacks of the Okamoto-Uchiyama cryptosystem is the fact that messages are bounded by p, but arithmetic is performed modulo $n = p^2q$. Thus, a ciphertext is approximately three times larger than the corresponding plaintext. It should be noted that this is still a drastic improvement over earlier systems, such as the GM cryptosystem. The bound on message space can pose a problem when the cryptosystem is used specifically for its homomorphic properties. Because p is part of the private key, a user cannot disclose precisely what the bound on messages is, which may be problematic if many encrypted messages are to be summed or multiplied.

Homomorphic Properties

The Okamoto-Uchiyama cryptosystem supports the homomorphic addition and subtraction of two ciphertexts, addition and multiplication by a known constant, and efficient re-randomization of ciphertexts. Given $c_0 = g^{m_0}h^{r_0}$ and $c_1 = g^{m_1}h^{r_1}$ as valid encryptions of m_0 and m_1 respectively,

$$c_0c_1 = g^{m_0+m_1}h^{r_0+r_1} \mod n$$

is a valid encryption of $m_0 + m_1$ with randomness $r_1 + r_2$. Multiplication by a constant k can be achieved by calculating

$$c_0{}^k = g^{km_0}h^{kr_0} \mod n$$

as a valid encryption of km_0 with randomness kr_0, and c_0 can be re-randomized by calculating

$$c_0h^r = g^{m_0}h^{rr_0} \mod n$$

for a random $r \in \mathcal{R}$. Subtraction can be achieved by multiplying $c_0c_1{}^{-1}$ and addition and subtraction by constants can be achieved by encrypting the constant under the public key and using one of the previously defined operations.

Variants

Coron, Naccache, and Paillier [36] have proposed a more efficient variant of the Okamoto-Uchiyama cryptosystem, reducing the complexity of decryption while retaining the same level of security. They note that if $p-1$ has a large prime factor t (i.e., about 160 bits in length), then h can be redefined by letting $p-1 = tu$, choosing a random generator $g' \in \mathbb{Z}_n^*$, and setting $H = g'^{nu} \mod n$. The new value H replaces h in Gen and Enc, and decryption becomes

$$m = \frac{L(c^t \mod p^2)}{L(g^{p-1} \mod p^2)} \mod p.$$

Thus, c is raised to the power t instead of $p-1$, where t is much smaller than $p-1$.

A different variant of the Okamoto-Uchiyama cryptosystem, referred to as Improved Okamoto-Uchiyama, has been proposed by Choi, Choi, and Won [31], which, like the variant proposed by Coron et al, decreases the complexity of decryption. This accomplished by choosing g such that $L(g^{p-1} \mod p) \equiv 1 \mod p$, simplifying decryption to $m = L(c^{p-1} \mod p^2)$. Let $b \in \mathbb{Z}_p$ and $c \in \mathbb{Z}_{p^2}$ such that $b \equiv \left(\frac{c^{p-1}-1}{p} - 1\right) c \mod p$, then set $g = bp + c \mod p^2$. Then g satisfies $L(g^{p-1} \mod p^2) \mod \equiv 1 \mod p$. Sakurai and Takagi [89] call this system Modified Okamoto-Uchiyama, and point out that it is still an open problem to demonstrate that encryption is not invertible when the choice of g is limited to the special form required in this variant. Semantic security of the Modified Okamoto-Uchiyama cryptosystem follows from the semantic security of the original cryptosystem.

Okamoto and Uchiyama attempted to define their cryptosystem over an elliptic curve with p points, where discrete logs are efficient to compute [78]; however, they find that a secure construction using their approach is not possible. Paillier later attempted to construct a similar cryptosystem based on the Okamoto-Uchiyama cryptosystem [80], but Galbraith has demonstrated a flaw in the approach [53] for which no workaround is currently known.

3.2.8 The Paillier Cryptosystem

The Okamoto-Uchiyama cryptosystem sparked further research into cryptosystems utilizing trapdoor discrete logarithms, leading to the discovery of the Paillier Cryptosystem [79]. Decrypting a ciphertext in the Paillier cryptosystem requires the use of a logarithm function L, similar to the Okamoto-Uchiyama cryptosystem, although it used slightly differently. Additionally, similar to the way the Benaloh cryptosystem extended the Goldwasser-Micali cryptosystem by utilizing the difficulty of calculating certain higher order residues instead of quadratic residues modulo a prime p, the Paillier cryptosystem extends the Benaloh cryptosystem by utilizing the difficulty of deciding higher order residues modulo a composite n^2, where $n = pq$.

Recall from Section 3.1.4 that the set of n'th residues of $\mathbb{Z}_{n^2}^*$ forms a subgroup of order $\phi(n)$, and each n'th residue has exactly n roots. The n'th residue classes of $\mathbb{Z}_{n^2}^*$ partition $\mathbb{Z}_{n^2}^*$ into $\phi(n)$ different classes, each of size n, and the class function $w \mapsto [w]_g$ maps an element of $\mathbb{Z}_{n^2}^*$ to one of these classes; that is, given $w = g^m y^n \in \mathbb{Z}_{n^2}$, the class function maps w to m. Because the computational problem of computing the class function, Class$[n]$, is thought to be hard, it is a natural building block for a cryptosystem. Additionally, the class function is homomorphic, meaning that any cryptosystem utilizing the class function as a decryption function will also be homomorphic.

Lemma 3.2.19. *The class function is a homomorphism from $\mathbb{Z}_{n^2}^*$ to $\mathbb{Z}_n$.*

Proof. Let $w_1, w_2, g \in \mathbb{Z}_{n^2}^*$. Then $[w_1]_g = x_1$ and $[w_2]_g = x_2$ and there exists y_1 and y_2 with $w_1 = g^{x_1}{y_1}^n$ and $w_2 = g^{x_2}{y_2}^n$. Set $y = y_1 y_2$, then $[w_1 w_2]_g = [w_1]_g + [w_2]_g = x_1 + x_2$ follows from the fact that $(g^{x_1}{y_1}^n)(g^{x_2}{y_2}^n) = g^{x_1+x_2}y^n$. □

The function $\mathcal{E}_g(x, y) = g^x y^n$ defined in Section 3.1.4 is a bijection from $\mathbb{Z}_n \times \mathbb{Z}_n^*$ to $\mathbb{Z}_{n^2}^*$ when the order of g is a multiple of n, and, when the order of g is αn for $\alpha \in \{1, \ldots, \lambda = \text{lcm}(p-1, q-1)\}$, g uniquely determines x for a given $w = g^x y^n$. Thus, $\mathcal{E}_g$ can take a message x and calculate w such that $[w]_g = x$, with y acting as a randomizer. The Paillier cryptosystem uses $\mathcal{E}_g$ as its encryption function, and the class function as its decryption function.

Cryptosystem 3.2.20. *The Paillier Cryptosystem*

Let $n = pq$ for primes p and q, set $\lambda = lcm(p-1, q-1)$, and choose $g \in \mathbb{Z}^*_{n^2}$ such that $\gcd(L(g^\lambda \mod n^2), n) = 1$*, where $L(x) = \frac{x-1}{n}$. Then $\mathcal{P} = \mathcal{R} = \mathbb{Z}_n$, $\mathcal{C} = \mathbb{Z}^*_{n^2}$, and $\mathcal{K} = \{(n, g, p, q, \lambda)\}$ where n, q, p, q, λ are defined as above.*

- *Gen: Given security parameter ϵ, $Gen(\epsilon)$ chooses two distinct $\frac{\epsilon}{2}$-bit primes p and q, sets $n = pq$ and $\lambda = lcm(p-1, q-1)$, and chooses $g \in \mathbb{Z}^*_{n^2}$ such that* $\gcd(L(g^\lambda \mod n^2), n) = 1$*. The tuple (n, g) is the public key, and the tuple (p, q, λ) is the private key.*

- *Enc: Given a message $m \in \mathcal{P}$ and a public key $pk = (n, g)$, $Enc(pk, m)$ chooses a random $r \in \mathcal{R}$ and returns the ciphertext*

$$c = g^m r^n \mod n^2.$$

- *Dec: Given a ciphertext $c \in \mathcal{C}$ and a private key $sk = (p, q, \lambda)$, $Dec(sk, c)$ returns the message*

$$m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n$$

  where

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

The security of the Paillier system relies on the hardness of the Class[n] and D-Class[n] problems, as defined in Section 3.1.4. Class[n] is the problem of determining which residue class in $\mathbb{Z}_{n^2}$ a ciphertext belongs to, while D-Class[n] is the problem of deciding if $[w]_g = m$ for a given m. Knowledge of the private key, in particular, $\lambda = \text{lcm}(p-1, q-1)$, allows one to solve Class[n]. Consider the value $L(w^\lambda \mod n^2)$ for $w \in \mathbb{Z}^*_{n^2}$, and let $g = 1 + n$. Then there exists x, y such that $w = g^x y^n \mod n^2$, and, $w^\lambda = g^{x\lambda} y^{n\lambda} \mod n^2$. Noting that $g^x = (1+n)^x = 1 + nx \mod n^2$ and λ is a multiple of the order of $\mathbb{Z}^*_n$, then $w^\lambda = g^{x\lambda} y^{n\lambda} = 1 + x\lambda n \mod n^2$. Thus,

$$L(w^\lambda \mod n^2) = \frac{1 + x\lambda n - 1}{n} = x\lambda = \lambda [w]_{1+n}.$$

To prove the correctness of decryption, the following lemma is required.

Lemma 3.2.21. *For $w, g_1, g_2 \in \mathbb{Z}^*_{n^2}$ where $|g_1|, |g_2| \in \{n, 2n, \ldots, \lambda n\}$. Then*

$$[w]_{g_1} = [w]_{g_2} [g_2]_{g_1}.$$

Proof. Let $[w]_{g_1} = x$, $[w]_{g_2} = x_1$, and $[g_2]_{g_1} = x_2$, Then there exists y with $w = g^x y^n$, as well as y_1 with $w = {g_2}^{x_1} {y_1}^n$, and y_2 with $g_2 = {g_1}^{x_2} {y_2}^n$. The lemma holds if $x = x_1 x_2$ and there exists y such that $w = g^x y^n$. Observe that

$$\begin{aligned} w &= {g_2}^{x_1} {y_1}^n \\ &= ({g_1}^{x_2} {y_2}^n)^{x_1} {y_1}^n \\ &= {g_1}^{x_1 x_2} ({y_2}^{x_1} y_1)^n. \end{aligned}$$

The result follows from setting $y = {y_2}^{x_1} y_1$. □

The decryption step of the Paillier cryptosystem simplifies to

$$\begin{aligned}\frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n &= \frac{\lambda[c]_{1+n}}{\lambda[g]_{1+n}} \mod n \\ &= \frac{[c]_g[g]_{1+n}}{[g]_{1+n}} \mod n \qquad \text{(by Lemma 3.2.21)} \\ &= [c]_g,\end{aligned}$$

yielding the message m. The security of the Paillier system is summarized in the following two theorems.

Theorem 3.2.22. *([79], Theorem 4) If Class[n] is hard in $\mathbb{Z}_{n^2}^*$ then the Paillier encryption function is not invertible by an adversary.*

Proof. Decrypting a ciphertext is, by definition, solving the computational composite residuosity problem. □

Theorem 3.2.23. *([79], Theorem 5) The Paillier cryptosystem is semantically secure if and only if D-Class[n] is hard.*

Proof. (Outline) Let m_0 and m_1 be two known messages and c a random encryption of either m_0 or m_1. If c is an encryption of m_0, then $c' = cg^{-m_0} = r^n$ is an n'th residue. Otherwise, if c is an encryption of m_1, c' is not an $n'th$ residue with overwhelming probability. If an adversary can decide n'th residues, then the adversary can check if c' is an n'th residue, thus distinguishing between encryptions of m_0 and m_1. Conversely, if an adversary can distinguish between encryptions of m_0 and m_1, then, given (w, x), the adversary must determine if there exists y such that $g^x y^n = w$, or, $y^n = g^{-x}w$, i.e., $g^{-x}w$ is an n'th residue. The adversary can choose a valid value of g and calculate $c = g^{m_0}g^{-x}w$, which is a valid ciphertext if $g^{-x}w$ is an n'th residue. The adversary's ability to distinguish between encryptions of m_0 and m_1 can be used to determine if c is a valid encryption of m_0, thus solving D-Class[n]. Hence, the semantic security of the Paillier cryptosystem is equivalent to solving D-Class[n]. □

Further investigation into the security of the Paillier cryptosystem has been made by Catalano, Gennaro, and Howgrave-Graham [23, 24], who have shown that the least significant bit of a message encrypted under the Paillier cryptosystem is a *hard-core bit*. Given a one-way function f and a predicate π, π is said to be a hard-core predicate on f if an adversary can guess the output of π with advantage only negligibly better than a random guess. In the case of a hard-core bit, the adversary cannot guess the value of the specified bit with non-negligible advantage over a random guess. The notion of a hard-core bit was introduced by Blum and Micali [13], and it was later shown by Goldreich and Levin [59] that any one-way function has a hard-core bit, although their proof does not disclose which bit is hard-core. Catalano et al further generalize their result to state that if computing the class c of w remains hard if the adversary is told $c < 2^b$, then the Paillier encryption function hides at least $n - b$ bits.

Homomorphic Properties

The Paillier cryptosystem supports homomorphic addition and subtraction of encrypted messages, as well as addition and subtraction of constants, and multiplication by a constant. Let $c_1 = g^{m_1} {y_1}^n \mod n^2$ and $c_2 = g^{m_2} {y_2}^n \mod n^2$. Then

$$c_1 c_2 \mod n^2 = g^{m_1} {y_1}^n g^{m_2} {y_2}^n = g^{m_1+m_2} y_1 {y_2}^n \mod n^2$$

is a valid encryption of $m_1 + m_2$,

$$c_1 g^k \mod n^2 = g^{m_1} {y_1}^n g^k = g^{m_1+k} {y_1}^n \mod n^2$$

is a valid encryption of $m_1 + k$, and

$${c_1}^k \mod n^2 = (g^{m_1} {y_1}^n)^k = g^{km_1} ({y_1}^k)^n \mod n^2$$

is a valid encryption of km_1. Subtraction of encrypted messages and constants can be accomplished by computing $c_1 {c_2}^{-1} \mod n^2$ and $c_1 g^{-k} \mod n^2$ respectively.

The Paillier cryptosystem also supports the re-randomization of messages such that if y' is a uniformly random element of $\mathcal{R}$, then

$$c_1 y'^n \mod n^2 = g^{m_1} {y_1}^n y'^n = g^{m_1} (y_1 y')^n \mod n^2$$

is a uniformly random valid encryption of m_1.

Variants

Along with the basic cryptosystem, Paillier also proposed a more efficient variant and a trapdoor permutation based on similar ideas.

Cryptosystem 3.2.24. *Paillier Trapdoor Permutation*

The setup of the Paillier trapdoor permutation is the same as Cryptosystem 3.2.20, with $\mathcal{P} = \mathbb{Z}_{n^2}$.

- *Gen: See Cryptosystem 3.2.20.*
- *Enc: Given m and a public key $pk = (n, g)$, $Enc(pk, m)$ first splits m into m_1 and m_2 such that $m = m_1 + nm_2$, then returns the ciphertext*

$$c = g^{m_1} {m_2}^n \mod n^2.$$

- *Dec: Given a ciphertext c and a private key sk, $Dec(sk, c)$ calculates*

$$m_1 = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n$$
$$c' = cg^{-m_1} \mod n$$
$$m_2 = c'^{n^{-1}} \mod n$$

and returns $m = m_1 + nm_2$.

The first step of decryption recovers m_1, just as in the original Paillier cryptosystem. Recovering m_2 involves taking an n'th root of $c' = {m_2}^n$. This is essentially the same as decrypting an RSA ciphertext, and is believed to be intractable as long as the factoring problem is hard. This places a restriction on the permutation; namely, that m_2 must be an element of $\mathbb{Z}_n^*$. Paillier's trapdoor permutation is malleable, although the possible operations do not appear to be useful. This is due to the fact that part of the message lies in an exponent, while the other portion is represented as an n'th root. Let $m = m_1 + nm_2$ and $m' = {m_1}' + n{m_2}'$ with encryptions c_1 and c_2 respectively. The product of c_1 and c_2 decrypts to

$$m_1 + m_1' + n(m_2 m_2') \mod n$$

and ${c_1}^k$ for some constant k decrypts to

$$m_1 + k + n({m_2}^k) \mod n.$$

Paillier has also proposed a more efficient variant of his cryptosystem. The modular exponentiation required during decryption can be made more efficient by altering the form of ciphertexts, although this also requires a different security assumption. Encryption also becomes more efficient by requiring only one modular exponentiation. Unlike the trapdoor permutation, the fast variant also preserves the same homomorphic operations as the basic Paillier cryptosystem.

Cryptosystem 3.2.25. *The Paillier Cryptosystem (Fast Decryption Variant)*

Let $\mathcal{P}, \mathcal{C}, \mathcal{R}, n, p, q, \lambda$ be as in Cryptosystem 3.2.20, and let $g \in \mathbb{Z}_{n^2}^*$ such that the order of g is αn for $\alpha \in \{1, \ldots, \lambda\}$. Then $\mathcal{K} = \{(n, g, p, q, \alpha)\}$.

- *Gen: Given security parameter ϵ, Gen(ϵ) chooses two distinct ϵ-bit primes p and q, sets $n = pq$ and $\lambda = lcm(p-1, q-1)$, and chooses $g \in \mathbb{Z}_{n^2}^*$ such that the order of g is αn for $\alpha \in \{1, \ldots, \lambda\}$. The tuple (n, g) is the public key, and the tuple (p, q, α) is the private key.*

- *Enc: Given a message m and a public key $pk = (n, g)$, Enc(pk, m) chooses a random $r \in \mathcal{R}$ and returns the ciphertext*

$$c = g^{m+nr} \mod n^2.$$

- *Dec: Dec: Given a ciphertext c and a private key $sk = (p, q, \lambda)$, Dec(sk, c) returns the message*

$$m = \frac{L(c^\alpha \mod n^2)}{L(g^\alpha \mod n^2)} \mod n$$

Paillier did not include a proof of security for the fast decryption variant in [79]. Proving that the system is semantically secure and that encryption is one-way is similar to the basic Paillier cryptosystem, but relies on the PDL$[n, g]$ and D-PDL$[n, g]$ problems defined in Section 3.1.4.

Theorem 3.2.26. *If PDL[n, g] is hard, then the encryption function of the Paillier fast decryption variant is not invertible by an adversary.*

Proof. Inverting the encryption function is, by definition, solving PDL$[n, g]$. $\square$

Theorem 3.2.27. *The fast decryption variant of the Paillier cryptosystem is semantically secure if and only if D-PDL[n, g] is hard.*

Proof. Let $m_0, m_1 \in \mathbb{Z}_n$ and let c be a random encryption of one of m_0 and m_1. Assume an adversary can determine if c is an encryption of m_0 or m_1. Then, given an instance of D-PDL$[n, g]$, the adversary must decide whether or not $[w]_g = x$. The adversary can construct the ciphertext $g^{m_0}g^{-x}w$, and, if $w = g^{x+rn}$, then $g^{m_0}g^{-x}w = g^{m_0+rn}$ and the adversary concludes that $[w]_g = x$ if the ciphertext is an encryption of m_0. Hence, if breaking semantic security is tractable, then solving D-PDL$[n, g]$ is tractable. Conversely, assume that the adversary can solve D-PDL$[n, g]$. If c is an encryption of m_0 then the adversary can solve D-PDL$[n, g]$ to determine if $[c]_g = m_0$. If it is not, then $[c]_g = m_1$. Thus, if the adversary can solve D-PDL$[g, n]$, then the adversary can break semantic security. Hence, the semantic security of the fast decryption variant is equivalent to solving D-PDL$[n, g]$. $\square$

An approach to increasing the efficiency of encryption was considered by Catalano, Gennaro, Howgrave-Graham, and Nguyen [25]. Recall Cryptosystem 1.3.3, the RSA cryptosystem, and that RSA encryption requires a single modular exponentiation modulo n. In order to make RSA encryption more efficient the public key is often selected to be a small value, such as $e = 3$. Catalano et al. argue that the same approach can be applied to the Paillier cryptosystem, by replacing y^n with y^e for some appropriate choice of $e << n$, with the security of the modified cryptosystem conjectured to hold for any $e > 2$. Additionally, encryption can be further simplified by choosing $g = 1+n$, so that $g^m = (1+mn) \mod n^2$. Thus, the encryption function becomes $c = g^m y^e = (y^e(1 + mn))$. The security assumptions for this variant are different than in the original cryptosystem, and are more related to RSA security assumptions. Proof that the encryption is one-way relies on the difficulty of the Computational Small e'th Root Problem (Section 3.1.2), and the proof of semantic security relies on the decisional version of the same problem. Castagnos [21] has recently modified the cryptosystem given by Catalano et al. in order to make it more efficient. The resulting cryptosystem achieves its goal, but does not retain any homomorphic properties. Castagnos does, however, outline an approach to making his cryptosystem homomorphic, although the details are omitted.

Paillier has proposed additional variants of his cryptosystem in subsequent publications. In [81] Paillier and Pointcheval extend the Paillier cryptosystem to provide IND-CCA2 security, necessarily eliminating its homomorphic properties. Paillier also proposed an elliptic curve version of his cryptosystem [80], attempting to take advantage of the fact that in certain settings computing discrete logs on elliptic curves is easy. Galbraith has since demonstrated that Paillier's elliptic curve cryptosystem is not secure [53], as the private key is revealed from public information. Galbraith also provided an approach to building an elliptic curve Paillier cryptosystem that does not suffer from the same problem as Paillier's approach. This cryptosystem is presented in Section 3.2.10.

As with the Okamoto-Uchiyama cryptosystem, Choi, Choi, and Won [31] have proposed a modified version of the Paillier cryptosystem, referred to as Modified Paillier, or the M-Paillier cryptosystem, by Sakurai and Takagi [89], who provided a security analysis of the proposed cryptosystem. Like the modified Okamoto-Uchiyama cryptosystem, M-Paillier is based on the idea of choosing g such that $L(g^\lambda \bmod n^2) \equiv 1 \mod n$, so that decryption can simplified to $m = L(c^\lambda \mod n^2) \bmod n$. This is accomplished by selecting g such that $g^\lambda = 1 + n \mod n^2$. The modification can be applied to both the basic and fast decryption variants of the Paillier cryptosystem.

Choi et al. did not provide any proofs of security for their modified cryptosystems, prompting an investigation by Sakurai and Takagi [89]. Because the possible values for g are limited the Class$[n]$ and D-Class$[n, g]$ problems must be restated to allow only certain values of g. When this is the case, Choi et al. show that solving Class$[n]$ is as hard as factoring. As a consequence, inverting the encryption function is as hard as factoring n, assuming that g can be generated from n alone without knowledge of p and q. If $g = 1+n$, then the appropriate assumption holds, however other choices of g are valid in the M-Paillier cryptosystem. In the case when $g \neq 1 + n$, then an oracle that generates public parameters can also factor n, i.e., the public parameters cannot be generated without knowledge of the factorization of n. The semantic security of M-Paillier holds as long as D-Class$[n, g]$ is hard on the restricted choices of g. Choi et al. also demonstrate a chosen-ciphertext attack, specific to M-Paillier, that recovers p and q using only a single decryption query.

3.2.9 Paillier/Schmidt-Samoa-Takagi Cryptosystem

Although the Paillier and Okamoto-Uchiyama cryptosystems are similar, the security of the Okamoto-Uchiyama cryptosystem can be reduced to factoring $n = p^2q$, while the Paillier cryptosystem relies on Class$[n]$. Because the factoring problem has been studied in more detail, Schmidt-Samoa and Takagi [91] have altered the Paillier cryptosystem such that it's security can be based on the difficulty of factoring $n = p^2q$. Unfortunately, this comes at the cost of introducing a chosen ciphertext attack, similar to the one that affects the Okamoto-Uchiyama cryptosystem. The basic Paillier cryptosystem is defined over $\mathbb{Z}_{n^2}$, which is isomorphic to $\mathbb{Z}_n^* \times \mathbb{Z}_n$ when $n = pq$, whereas the proposed variant is defined over $\mathbb{Z}_n^* \times \mathbb{Z}_{pq}$ when $n = p^2q$.

Cryptosystem 3.2.28. *Paillier Cryptosystem (Schmidt-Samoa-Takagi Variant)*

Let p, q be distinct primes such that both $p-1$ and $q-1$ have a large prime factor, $p \not| q-1$, and $q \not| p-1$. Choose l such that $2^l < pq < 2^{l+1}$. Then $\mathcal{P} = \{0,1\}^l$, $\mathcal{C} = \mathbb{Z}_{n^2}^*$, $\mathcal{R} = \mathbb{Z}_n^*$, and $\mathcal{K} = \{(n, l, d, p, q) \mid d = n^{-1} \mod (p-1)(q-1)\}$.

- *Gen: Given a security parameter ϵ, Gen(ϵ) chooses two distinct $\frac{\epsilon}{3}$-bit primes p and q and sets $n = p^2q$, then chooses l such that $2^l < pq < 2^{l+1}$ and d such that $d = n^{-1} \mod (p-1)(q-1)$. The public key is (n, l), and the private key is (d, p, q).*

- *Enc: Given a public key pk and a message $m \in \mathcal{P}$, Enc(pk, m) chooses $r \in \mathcal{R}$ and calculates the ciphertext*
$$r^n(1 + mn) \mod n^2.$$

- *Dec: Given a private key sk and a ciphertext $c \in \mathcal{C}$, Dec(sk, c) calculates $r = c^d \mod pq$, and then the message*
$$m = L(r^{-n}c \mod n^2) \mod pq$$
where
$$L(x) = \frac{x-1}{n}.$$

Like other Paillier variants, the Schmidt-Samoa-Takagi cryptosystem chooses $g = 1 + n$ such that $g^m \mod n^2 = (1 + nm) \mod n^2$, thus making encryption more efficient. During decryption the random factor r is recovered by taking

$$\begin{aligned}
c^d \mod n &= (r^n(1+mn))^d \mod pq \\
&= (r^n)^d \mod pq \\
&= (r^n)^{n^{-1} \mod (p-1)(q-1)} \mod pq \\
&= (r^n)^{n^{-1} \mod \phi(pq)} \mod pq \\
&= r
\end{aligned}$$

allowing the random factor to be removed from the ciphertext. Recovering r is essentially an RSA decryption, and requires knowledge of the factorization of pq. With the randomness removed from the ciphertext, recovering the message is easy using the L function.

Given access to a decryption oracle, a method for factoring n with non-negligible probability is possible.

Theorem 3.2.29. *([89], Theorem 3) Inverting the Schmidt-Samoa-Takagi encryption function is equivalent to factoring $n = p^2q$.*

Proof. (Outline.) Factoring n trivially allows the decryption of a ciphertext. Assume that an oracle exists that can decrypt ciphertexts with non-negligible probability. Then, with non-negligible probability, a random ciphertext $c' = r'^n(1 + m'n)$ will be valid for $r' \in \mathbb{Z}_n^*$ and $m \in \mathbb{Z}_n$, and the decryption oracle will return a message m such that there exists $r \in \mathbb{Z}_n^*$ with $c' = r^n(1 + mn)$, and $m \equiv m' \mod pq$. By taking $\gcd(m' - m, n)$, the factor pq is recovered, allowing n to be factored with non-negligible probability. Hence, inverting the decryption function is equivalent to factoring. □

As with the Okamoto-Uchiyama cryptosystem, the fact that inverting the encryption function is equivalent to factoring n can be exploited by an adaptive adversary to recover the private key. Given access to a decryption oracle, the adversary chooses random $r' \in \mathbb{Z}_n^*$ and $m' \in \mathbb{Z}_n$ and constructs the ciphertext $c' = r'^n(1 + m'n)$, which is submitted to the decryption oracle. The adversary then learns m such that $m \equiv m' \mod pq$, and, with non-negligible probability, $\gcd(m' - m, n) = pq$. Thus, the adversary can factor n and recover the private key.

The Schmidt-Samoa-Takagi variant inherits semantic security from the Paillier cryptosystem.

Theorem 3.2.30. *([89], Theorem 4) The Schmidt-Samoa-Takagi variant is semantically secure if and only if D-Class[n] is hard when $n = p^2q$.*

Proof. The proof is the same as Theorem 3.2.23 with $n = p^2q$ and $g = (1 + n)$. □

The Schmidt-Samoa-Takagi variant also inherits the homomorphic properties of the Paillier cryptosystem, as detailed in Section 3.2.8.

3.2.10 Elliptic Curve Paillier

Galbraith [53] demonstrated that Paillier's elliptic curve cryptosystem, defined in [80], is not secure, as it suffers from the same problem Okamoto and Uchiyama encountered attempting to create an elliptic curve variant of their cryptosystem [78]; namely, that the use of elliptic curves where discrete logs are easy, so-called anomalous curves, allows the private key to be recovered from the necessary public information. In response, Galbraith has provided some ideas on how to build a Paillier-like cryptosystem over an elliptic curve, with a concrete example given.

Cryptosystem 3.2.31. *Elliptic Curve Paillier*

Let $n = pq$ for distinct odd primes p and q, and let $E = x^3 + axz^2 + bz^3$ be a random elliptic curve over $\mathbb{Z}_n$ such that $\gcd(6(4a^3 + 27b^2), n) = 1$. Set $M = lcm(|E(\mathbb{F}_p)|, |E(\mathbb{F}_q)|)$ and $Q = nQ'$ where Q' is a random point on the curve over $\mathbb{Z}_{n^2}$. Let $P_m = (mn : 1 : 0)$ be the point on the curve $E(\mathbb{Z}_{n^2}$ corresponding to $m \in \mathbb{Z}_n$, then $\mathcal{P} = \{P_m \,|\, m \in \mathbb{Z}_n\}$, $\mathcal{C} = E(\mathbb{Z}_{n^2})$, $\mathcal{R} = \mathbb{Z}_n - \{0\}$, and $\mathcal{K} = \{(n, a, b, Q, M)\}$ such that n, a, b, Q, M satisfy the above conditions.

- *Gen: Given a security parameter ϵ, Gen(ϵ) chooses two distinct $\frac{\epsilon}{2}$-bit primes p and q, and sets $n = pq$. A random elliptic curve $E = x^3 + axz^2 + bz^3$ over $\mathbb{Z}_n$ is chosen such that $\gcd(6(4a^3 + 27b^2), n) = 1$ and M is set to $lcm(|E(\mathbb{F}_p)|, |E(\mathbb{F}_q)|)$. Gen($\epsilon$) then chooses a random point Q' on the curve and sets $Q = nQ'$. The public key is (a, b, n, Q) and the private key is M.*

- *Enc: Given a public key pk and a message m, Enc(pk, m) chooses a random $r \in \mathcal{R}$ and calculates the ciphertext*

$$c = rQ + P_m.$$

- *Dec: Given a private key sk and a ciphertext c, $Dec(sk, c)$ calculates*

$$Mc = MrQ + MP_m = P_{mM} = (mM : 1 : 0)$$

and returns the message

$$m = mMM^{-1}.$$

The components of Galbraith's elliptic curve variant match those of the basic Paillier cryptosystem. The message and ciphertext spaces are replaced with the analogous elliptic curve groups, λ is replaced by M, the least common multiple of the number of points of the elliptic curve in $\mathbb{F}_p$ and $\mathbb{F}_q$, and g is replaced with Q, a random point whose order divides M. The encryption function takes a message as a point on the curve, and "blinds" it with a random multiple of Q, similar to the way a message is randomized using y^n in the basic cryptosystem. Multiplying a ciphertext by the private key M eliminates the random factor and allows the message to be recovered.

Galbraith does not prove the security of his variant, but notes that the semantic security of the system relies on the problem of deciding whether or not a random point S lies in the subgroup of $E(\mathbb{Z}_{n^2})$ generated by Q. This is reminiscent of the p-subgroup problem, upon which the security of the Okamoto-Uchiyama cryptosystem relies. Without knowledge of the factorization of n, there is no known approach to solving the problem in general.

3.2.11 The Paillier/Damgård-Jurik Cryptosystem

The Paillier cryptosystem is generally regarded as the most practical homomorphic cryptosystem. The encryption function is of the same form as earlier additively homomorphic cryptosystems, and decryption is essentially a modular exponentiation. One of the biggest benefits of the Paillier cryptosystem is the fact that it achieves a 2-to-1 ciphertext expansion ratio, while retaining straightforward encryption and decryption methods. The Damgård-Jurik cryptosystem [40] extends the Paillier cryptosystem by taking messages from $\mathbb{Z}_{n^s}$ and mapping them to elements $\mathbb{Z}^*_{n^{s+1}}$, thus achieving a ciphertext expansion rate of $\frac{s+1}{s}$, which can be made arbitrarily close to 1 for large enough s. This comes with the negative aspect of increasing the size of the modulus, thereby slowing down modular arithmetic and increasing the minimum message size.

The Paillier cryptosystem utilized the fact that $\mathbb{Z}^*_{n^2}$ is isomorphic $\mathbb{Z}_n \times \mathbb{Z}^*_n$. The Damgård-Jurik cryptosystem naturally extends this by noting that $\mathbb{Z}^*_{n^{s+1}}$ is isomorphic to $G \times H$ where $G = \mathbb{Z}_{n^s}$ and $H = \mathbb{Z}^*_n$. Thus, the factor group $\frac{\mathbb{Z}_{n^{s+1}}}{H}$ is isomorphic to $\mathbb{Z}_{n^s}$.

Many other variants of the Paillier cryptosystem choose g of the form $g = (1+n)^k$ due to the fact that $(1 + n)$ has order n and $(1 + n)^k \mod n^2 = (1 + kn) \mod n^2$ is easy to compute. In the Damgård-Jurik cryptosystem, the same choice of g can be used.

Lemma 3.2.32. *([40], Lemma 1) For $s < p, q$, the element $(1+n)$ has order n^s in $\in \mathbb{Z}_{n^{s+1}}$.*

The order of H, $|H| = \phi(n)$ is relatively prime to n^s, and so $(1+n)H$ is a generator of $\frac{\mathbb{Z}_{n^{s+1}}}{H}$, and elements of the factor group can be represented by

$$H, (1+n)H, (1+n)^2H, \ldots, (1+n)^{n^s-1}H.$$

These cosets form the plaintext space of the Damgård-Jurik cryptosystem, where $(1+n)^m H$ is the coset element representing the message $m \in \mathbb{Z}_{n^s}$. Recalling the decryption function from the Paillier cryptosystem (Cryptosystem 3.2.20), in order to recover a message the value $L(c^\lambda \mod n^2)$ is required. Because messages are encoded as cosets, a method for recovering i from $L((1+n)^i \mod n^{s+1})$ will be required to recover messages during decryption. Damgård and Jurik provide the following algorithm for recovering i.

```
.   $i := 0$
.   for $j := 1$ to $s$ do begin
.       $t_1 := L(a \mod n^{j+1})$
.       $t_2 := i$
.       for $k := 2$ to $j$ do begin
.           $i := i - 1$
.           $t_2 := t_2 * i \mod n^j$
.           $t_1 := t_1 - \frac{t_2 n^{k-1}}{k!} \mod n^j$
.       end
.       $i := t_1$
.   end
```

The algorithm works by calculating $i \mod n^j$ from

$$L((1+n)^i \mod n^{s+1}) = \prod_{k=1}^{n^s} \left(\binom{i}{1} n^{k-1} \right) \mod n^s$$

for successive values of j until $i \mod n^s$ is recovered.

Cryptosystem 3.2.33. *The Damgård-Jurik Cryptosystem*

Let $n = pq$ for distinct odd primes p and q, and let $G \times H = \mathbb{Z}^*_{n^{s+1}}$ where H is isomorphic to $\mathbb{Z}^*_n$. Set $\lambda = lcm(p-1, q-1)$ and choose $g = (1+n)^j x \mod n^{s+1}$ such that $\gcd(j, n) = 1$ and $x \in H$. Then $\mathcal{P} = \mathbb{Z}_{n^s}$, $\mathcal{C} = \mathcal{R} = \mathbb{Z}^*_{n^{s+1}}$, and $\mathcal{K} = \{(n, s, g, j, \lambda)\}$ where n, s, g, j, λ are defined as above.

- *Gen: Given a security parameter ϵ, Gen(ϵ) chooses two $\frac{\epsilon}{2}$-bit primes p and q, then sets $n = pq$, $\lambda = lcm(p-1, q-1)$, and $g = (1+n)^j x \mod n^{s+1}$ such that $\gcd(n, j) = 1$ and x is a random element of H. The public key is (n, s, g) and the private key is (j, λ).*

- *Enc: Given a public key pk and a message $m \in \mathcal{P}$, Enc(pk, m) chooses a random $r \in \mathcal{R}$ and calculates the ciphertext*

$$c = g^m r^{n^s} \mod n^{s+1}.$$

- *Dec: Given a private key sk and a ciphertext $c \in \mathcal{C}$, Dec(sk, c) calculates*

$$c^\lambda \mod n^{s+1} = (1+n)^{mj\lambda \mod n^s}$$

and

$$g^\lambda \mod n^{s+1} = (1+n)^{j\lambda \mod n^s}$$

and uses the algorithm for recovering k from $L((1+n)^k \mod n^{s+1})$ to calculate $mj\lambda$ and $j\lambda$. The message is calculated by

$$m = \frac{mj\lambda}{j\lambda} \mod n^s.$$

Although presented slightly differently, the decryption algorithm calculates

$$\frac{L(c^\lambda \mod n^{s+1})}{L(g^\lambda \mod n^{s+1})} \mod n^s$$

as in the original Paillier cryptosystem. Note that

$$\begin{aligned} c^\lambda \mod n^{s+1} &= (g^m r^{n^s})^\lambda \mod n^{s+1} \\ &= (g^m)^\lambda (r^{n^s})^\lambda \mod n^{s+1} \\ &= (1+n)^{mj\lambda} (xr^{n^s})^\lambda \mod n^{s+1} \\ &= (1+n)^{mj\lambda} \mod n^{s+1}, \end{aligned}$$

due to the fact that $(xr^{n^s})^\lambda \equiv 1 \mod n^{s+1}$.

The security of the Damgård-Jurik Cryptosystem follows from the security of the Paillier cryptosystem.

Theorem 3.2.34. *If Class[n] is hard for higher powers of n, then the Damgård-Jurik encryption function is not invertible by an adversary.*

Proof. Recall Definition 3.1.12, the definition of Class$[n, g]$. If the problem is redefined over $\mathbb{Z}^*_{n^{s+1}}$, then inverting the Damgård-Jurik encryption function is, by definition, solving the computational problem Class$[n]$ over $\mathbb{Z}^*_{n^{s+1}}$. □

Similarly, the semantic security relies on the same assumption as the Paillier cryptosystem.

Theorem 3.2.35. *The Damgård-Jurik cryptosystem is semantically secure if and only if D-Class[n] is hard.*

Proof. (Outline) Let c_s be a ciphertext modulo n^{s+1}. If $s = 1$ then the semantic security of the Damgård-Jurik cryptosystem follows from the security of the Paillier cryptosystem. If an adversary can solve D-Class$[n]$ with non-negligible probability, then for $s > 1$ the adversary can reduce a ciphertext modulo n^2 to obtain a ciphertext from the basic Paillier cryptosystem. Thus, with non-negligible probability an adversary that can solve D-Class$[n]$ can break the semantic security of the Damgård-Jurik cryptosystem. The other direction can be proved inductively using the security of the Paillier cryptosystem when $s = 1$ as a starting point. □

Damgård and Jurik also provided a variant of their cryptosystem that allows the block length to be decided at the time of encryption rather than as a fixed parameter, a property referred to as *length-flexible*. The public key can also be simplified by fixing $g = 1 + n$, with no impact on the security of the cryptosystem.

Cryptosystem 3.2.36. *The Damgård-Jurik Cryptosystem (Length-Flexible Variant)*

Let $n = pq$ for distinct odd primes p and q, and set $\lambda = lcm(p-1, q-1)$. Then $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{R}$ are as in Cryptosystem 3.2.33, and $\mathcal{K} = \{(n, \lambda)\}$.

- *Gen: Given a security parameter ϵ, $Gen(\epsilon)$ chooses two $\frac{\epsilon}{2}$-bit primes p and q, then sets $n = pq$ and $\lambda = lcm(p-1, q-1)$. The public key is n and the private key is λ.*

- *Enc: Given a public key pk and a message $m \in \mathcal{P}$, $Enc(pk, m)$ chooses a block size s such that $m \in \mathbb{Z}_{n^s}$ and chooses a random $r \in \mathbb{Z}^*_{n^{s+1}}$, then calculates the ciphertext*
$$c = (1+n)^m r^{n^s} \mod n^{s+1}.$$

- *Dec: Given a private key sk and a ciphertext $c \in \mathcal{C}$, $Dec(sk, c)$ chooses s such that $c \in \mathbb{Z}_{n^{s+1}}$ and calculates*
$$c^\lambda \mod n^{s+1} = (1+n)^{m \mod n^s}$$
and uses the algorithm for recovering k from $L((1+n)^k \mod n^{s+1})$ to recover m.

The security of the adjustable block length variant follows from the security of the basic cryptosystem, although a complete proof of a similar variant, using a different approach, has also been given by Damgård and Jurik [41, 42]. This variant modifies the cryptosystem in such a way that multiple users may use the same public modulus, while retaining the homomorphic properties of the original Paillier cryptosystem. This is accomplished by using a private key that does not rely on the factorization of the modulus.

Cryptosystem 3.2.37. *The Modified Length-Flexible Damgård-Jurik Cryptosystem*

Let $n = pq$ for distinct off primes p and q, where $p = 2p' + 1$ and $q = 2q' + 1$ for distinct odd primes p' and q', and set $m = p'q'$. Let g be an element of the subgroup of squares of $\mathbb{Z}_n^*$ and α be a random element of $\mathbb{Z}_m$. Then $\mathcal{P} = \mathbb{Z}^+$, $\mathcal{C} = \mathbb{Z}^+ \times \mathbb{Z}^+$, $\mathcal{R} = \mathbb{Z}_n$, and $\mathcal{K} = \{n, g, \alpha, h = g^\alpha \mod n\}$ where n, g, and α are as defined.

- *Gen: Given a security parameter ϵ, Gen(ϵ) chooses two distinct $\frac{\epsilon}{4}$-bit primes p' and q' such that $p = 2p' + 1$ and $2q' + 1$ are also prime, and sets $m = pq$. Gen(ϵ) then chooses $g \in \mathbb{Z}_n^*$ and $\alpha \in \mathbb{Z}_m$ at random such that g is a square in $\mathbb{Z}_n^*$. The public key is (n, g, h) and the private key is α.*

- *Enc: Given a public key pk and a message M, Enc(pk, M) chooses s such that $M \in \mathbb{Z}_{n^s}$ and a random $r \in \mathcal{R}$ and calculates the ciphertext*

$$c = (G, H) = (g^r \mod n, (h^r \mod n)^{n^s}(1 + n)^m \mod n^{s+1}).$$

- *Dec: Given a private key sk and a ciphertext $c = (G, H)$, Dec(sk, c) selects s such that $G, H \in \mathbb{Z}_{n^s}$, then calculates*

$$\begin{aligned} M' &= (H(G^\alpha \mod n)^{-n^s}) \\ &= (h^r \mod n)^{n^s}(1+n)^m \mod n^{s+1}(g^\alpha r \mod n)^{-n^s} \\ &= (1+n)^m \mod n^{s+1}. \end{aligned}$$

  and applies the algorithm presented earlier on M' to recover M.

Once an instance of the modified length-flexible variant is created, additional instances can be generated using the same modulus by selecting $\alpha \in \mathbb{Z}_n$ and setting $h = g^\alpha$. Although α should be selected uniformly from $\mathbb{Z}_m$, which requires knowledge of m, and hence the factorization of the modulus, Damgård and Jurik note that the values of h produced by choosing $\alpha \in \mathbb{Z}_n$ will be computationally indistinguishable from those produced by choosing $\alpha \in \mathbb{Z}_m$.

The security of the modified length-flexible variant is summarized in the following theorem.

Theorem 3.2.38. *([42], Theorem 1) Assuming the decisional composite residuosity problem (Section 3.1.4)and decisional Diffie-Hellman problem (Section 3.1.6) are hard, the modified length-flexible Damgård-Jurik cryptosystem is semantically secure.*

Damgård and Jurik also note that careful choice of parameters allows the security of the cryptosystem to rely only on the decisional composite residuosity problem; however, the proof relying on both assumptions is provided in [41].

The message expansion in the Damgård-Jurik cryptosystems is $\frac{s+1}{s}$, with modular arithmetic becoming slower as s grows. In the adjustable block length variant, the person encrypting the message can choose a smaller value of s if the message is small, thus making modular arithmetic more efficient when possible. The Damgård-Jurik cryptosystem is of additional interest, as it supports an efficient threshold variant, such that a dealer can distribute the private key to a group of l players such that at least t of them must cooperate in order to decrypt a message. The threshold variant is presented in Chapter 4.

Homomorphic Properties

The Damgård-Jurik cryptosystems inherit all the homomorphic properties of the Paillier cryptosystem. Let $c_1 = g^{m_1}{r_1}^n \mod n^{s+1}$ and $c_2 = g^{m_2}{r_2}^n \mod n^{s+1}$. Then

$$c_1c_2 \mod n^{s+1} = g^{m_1}{r_1}^n g^{m_2}{r_2}^n = g^{m_1+m_2}r_1{r_2}^n \mod n^{s+1}$$

is a valid encryption of $m_1 + m_2$,

$$c_1g^k \mod n^{s+1} = g^{m_1}{r_1}^n g^k = g^{m_1+k}{r_1}^n \mod n^{s+1}$$

is a valid encryption of $m_1 + k$, and

$${c_1}^k \mod n^{s+1} = (g^{m_1}{r_1}^n)^k = g^{km_1}({r_1}^k)^n \mod n^{s+1}$$

is a valid encryption of km_1. Subtraction of encrypted messages and constants can be accomplished by computing $c_1{c_2}^{-1} \mod n^{s+1}$ and $c_1g^{-k} \mod n^{s+1}$ respectively.

Re-randomization of messages is possible such that if r' is a uniformly random element of $\mathcal{R}$, then

$$c_1y'^n \mod n^{s+1} = g^{m_1}{r_1}^n r'^n = g^{m_1}(r_1r')^n \mod n^{s+1}$$

is a uniformly random valid encryption of m_1.

3.2.12 The Boneh-Goh-Nissim Cryptosystem

All of the additively homomorphic cryptosystems considered this far gain their homomorphic properties by encrypting messages “in the exponent”, such that the product of two ciphertexts has the result of summing the two plaintext messages in the exponent. This has the side effect of limiting homomorphic multiplication to known constants. The Boneh-Goh-Nissim cryptosystem utilizes a bilinear pairing to allow the computation of a single homomorphic multiplication of two ciphertexts, while still retaining the additive homomorphic properties of earlier cryptosystems.

Recall the definition of a bilinear pairing from Section 3.1.7. If e is a bilinear pairing, then, for a group element α,

$$e(\alpha^a, \alpha^b) = e(\alpha, \alpha)^{ab}$$

by the bilinear property. If an additively homomorphic cryptosystem is defined over a group that supports a bilinear pairing, then the bilinear property of the pairing can be used to take the product of the exponent of the ciphertext, i.e., multiply two messages together.

Boneh, Goh, and Nissim [14] observed this fact, and defined a Paillier-like cryptosystem over an elliptic curve group $\mathbb{G}$ supporting a bilinear pairing to an isomorphic group $\mathbb{G}_1$, that computes the homomorphic multiplication of two encrypted messages while preserving the additive homomorphic properties of the cryptosystem. Thus, an unlimited number of homomorphic additions may be performed, followed by a single multiplication, followed by an unlimited number of additions.

The method used by Boneh et al. to generate $\mathbb{G}$ and $\mathbb{G}_1$ is described in Section 3.1.7, and is based on the modified Weil pairing [73].

Cryptosystem 3.2.39. *Boneh-Goh-Nissim (BGN) Cryptosystem*

Let $n = pq$ for distinct odd primes p and q, let $\mathbb{G}, \mathbb{G}_1$ be two multiplicative groups of order n with a bilinear pairing $e : (\mathbb{G} \times \mathbb{G}) \to \mathbb{G}_1$, let g be a random generator of $\mathbb{G}$, and let h be a random generator of the subgroup of $\mathbb{G}$ of order p. Let $T < q$, then $\mathcal{P} = \mathbb{Z}_T$, $\mathcal{C} = \mathbb{G}$, $\mathcal{R} = \mathbb{Z}_n$, and $\mathcal{K} = \{(n, p, q, T, \mathbb{G}, \mathbb{G}_1, e, g, h)\}$ where $(n, p, q, T, \mathbb{G}, \mathbb{G}_1, e, g, h)$ are defined as above.

- *Gen: Given a security parameter ϵ, Gen(ϵ) chooses two distinct $\frac{\epsilon}{2}$-bit primes p, q, sets $n = pq$ and selects a positive integer $T < q$. Using the method described in Section 3.1.7, Gen(ϵ) then chooses two multiplicative groups $\mathbb{G}, \mathbb{G}_1$ of order n, that support a bilinear pairing $e : (\mathbb{G} \times \mathbb{G}) \to \mathbb{G}_1$, as well as random generators $g, u \in \mathbb{G}$, and sets $h = u^q$ such that h is a generator of the subgroup of order p. The public key is $(n, g, h, \mathbb{G}, \mathbb{G}_1, e)$, and the private key is p.*

- *Enc: Given a message $m \in \mathcal{P}$ and a public key pk, Enc(pk, m) chooses a random $r \in \mathcal{R}$ and calculates the ciphertext*

$$c = g^m h^r \mod n.$$

- *Dec: Given a ciphertext $c \in \mathcal{C}$ and a private key sk, Dec(sk, c) calculates*

$$c' = c^p = (g^p)^m \mod n$$

and uses Pollard's lambda method to take the discrete logarithm of c' in base g^p.

During decryption, c' is calculated as

$$\begin{aligned} c' &= c^p \mod n \\ &= (g^m h^r)^p \mod n \\ &= (g^m)^p \mod n \\ &= (g^p)^m \mod n \end{aligned}$$

due to the fact that $h^p \equiv 1 \mod n$. The message m is bounded by T, allowing it to be recovered in time $O(\sqrt{T})$ using one of the "square-root" discrete algorithms from Section 3.1.6, such as Pollard's lambda method.

The security of the BGN cryptosystem is summarized in the following theorem.

Theorem 3.2.40. *([14], Theorem 3.1) Let $n = pq$ for distinct odd primes p and q. The BGN cryptosystem is semantically secure if deciding whether or not a random element of $\mathbb{G}$ has order p is hard when p and q are unknown.*

Boneh, Goh, and Nissim call the problem of deciding whether or not a random element is a member of the subgroup of order p the *subgroup decision problem*, and show that an attacker that can break the semantic security of the BGN cryptosystem can solve a random instance of the subgroup decision problem.

Homomorphic Properties

Ciphertexts in the BGN cryptosystem are similar to the Paillier and Okamoto-Uchiyama cryptosystems, and inherit the same homomorphic properties. Let $c_1 = g^{m_1}h^{r_1} \mod n$ and $c_2 = g^{m_2}h^{r_2} \mod n$. Then

$$c_1c_2 \mod n = g^{m_1}h^{r_1}g^{m_2}h^{r_2} = g^{m_1+m_2}h^{r_1+r_2} \mod n$$

is a valid encryption of $m_1 + m_2$,

$$c_1g^k \mod n = g^{m_1}h^{r_1}g^k = g^{m_1+k}h^{r_1} \mod n$$

is a valid encryption of $m_1 + k$, and

$$c_1{}^k \mod n = (g^{m_1}h^{r_1})^k = g^{km_1}h^{r_1k} \mod n$$

is a valid encryption of km_1. Subtraction of encrypted messages and constants can be accomplished by computing $c_1c_2{}^{-1} \mod n$ and $c_1g^{-k} \mod n$ respectively.

Re-randomization of messages is possible such that if r' is a uniformly random element of $\mathcal{R}$, then

$$c_1h^{r'} \mod n = g^{m_1}h^{r_1}h^{r'} = g^{m_1}h^{r_1+r'} \mod n$$

is a uniformly random valid encryption of m_1.

In addition to the usual additive homomorphic operations, the BGN cryptosystem also allows a single homomorphic multiplication of plaintexts. Using the bilinear pairing e, set $g_1 = e(g, g)$ and $h_1 = e(g, h)$. The bilinear pairing is used to map two ciphertexts in $\mathbb{G}$ to the ciphertext representing the product of the two messages, in the isomorphic group $\mathbb{G}_1$, where g and h are replaced by g_1 and h_1 respectively. Recall that, because g generates $\mathbb{G}$, it holds that $h = g^\alpha$ for some α.

Given c_1, c_2, and a random $r \in \mathcal{R}$, a ciphertext representing the product $m_1 m_2$ can be calculated by

$$\begin{aligned}
e(c_1, c_2){h_1}^r &= e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}){h_1}^r \\
&= e(g^{m_1} g^{\alpha r_1}, g^{m_2} g^{\alpha r_2}){h_1}^r \\
&= e(g^{m_1 + \alpha r_1}, g^{m_2 + \alpha r^2}){h_1}^r \\
&= e(g, g)^{(m_1 + \alpha r_1)(m_2 + \alpha r_2)}{h_1}^r \\
&= {g_1}^{m_1 m_2 + m_1 \alpha r_2 + m_2 \alpha r_1 + \alpha^2 r_1 r_2}{h_1}^r \\
&= {g_1}^{m_1 m_2}{h_1}^{m_1 r_2 + m_2 r_1 + \alpha r_1 r_2 + r} \\
&= {g_1}^{m_1 m_2}{h_1}^{\bar{r}}
\end{aligned}$$

where $\bar{r}$ is a uniformly random element of $\mathcal{R}$. By replacing g and h with g_1 and h_1 respectively, the other homomorphic operations are still possible on the resulting ciphertext. Further multiplications are not possible as there is no pairing defined from $\mathbb{G}_1$ to another isomorphic group.

3.2.13 Insecure Homomorphic Cryptosystems

For completeness, some proposed homomorphic cryptosystems that have been subsequently been shown to be insecure are described.

Domingo-Ferrer has proposed two different privacy homomorphisms. The first [44] is based on using a secret sharing scheme to split an encrypted message into a vector of shares, which are then multiplied by secret values. To recover the message, each component is multiplied by the inverse of the secret values and Chinese remaindering is used to reconstruct the message. The corresponding cryptanalysis is given by Cheon, Kim, and Nam [28]. Domingo-Ferrer's second privacy homomorphism [45] is similar in structure to the first, and was subsequently broken by Wagner [102], and Bao [6].

Grigoriev and Ponomarenko [63] have proposed a homomorphic cryptosystem that maps an arbitrary finite group, given in terms of generators and relations, to a subset of the special linear group on 2×2 matrices. Choi [32], and Choi, Blackburn, and Wild [33] have demonstrated that the private key in this system can be recovered from the public key, thus rendering it insecure.

3.3 A Summary of Homomorphic Cryptosystems

Although the cryptosystems presented in this chapter have been defined over a variety of different groups, for comparison purposes the following notation will be adopted:

- n - The size of the ciphertext space; such that $\log_2 n \geq 1024$ and the factoring problem on n is hard.
- m - The size of the message space.
- k - Soundness parameter; chosen such that 2^{-k} is an acceptable error probability.
- s - Exponent used in Damgård-Jurik and variants.
- r - A bound on message size to allow for efficient discrete logarithm computation on exponents bounded by r.
- $\boxtimes, \boxplus$ - Homomorphic multiplication and addition on ciphertexts. $\boxtimes_c$ and $\boxplus_c$ will be used to denote multiplication and addition by a known constant. $\boxminus$ will be used to denote homomorphic subtraction.

Cryptosystem	Security Assumption	Homomorphic Operations	Message Expansion
RSA	RSA Problem	$\boxtimes$	1
Goldwasser-Micali	Quadratic Residuosity Problem	XOR	n
ElGamal	CDH / DDH	$\boxtimes$	2
Benaloh	Weak r'th Root Problem	$\boxplus, \boxminus, \boxtimes_c$	$\frac{n}{r}$
Naccache-Stern	Factoring / DLP / Weak r'th Residue Problem	$\boxplus, \boxminus, \boxtimes_c$	≥ 4
Sander-Young-Yung	Quadratic Residuosity Problem	AND	kn
Okamoto-Uchiyama	Factoring / p-subgroup Problem	$\boxplus, \boxminus, \boxtimes_c$	3
Modified Okamoto-Uchiyama	Factoring / p-subgroup Problem	$\boxplus, \boxminus, \boxtimes_c$	3
Improved Okamoto-Uchiyama	Factoring / p-subgroup Problem	$\boxplus, \boxminus, \boxtimes_c$	3
Paillier	Class$[n]$ / D-Class$[n]$	$\boxplus, \boxminus, \boxtimes_c$	2
Fast Decryption Paillier	PDL$[n]$ / D-PDL$[n]$	$\boxplus, \boxminus, \boxtimes_c$	2
Small Exponent Paillier	Small e'th Root Problem	$\boxplus, \boxminus, \boxtimes_c$	2
Modified Paillier	Class$[n]$ / D-Class$[n]$ on restricted generators	$\boxplus, \boxminus, \boxtimes_c$	2
Schmidt-Samoa-Takagi	Factoring Problem	$\boxplus, \boxminus, \boxtimes_c$	3
Elliptic Curve Paillier	Subgroup Decision Problem	$\boxplus, \boxminus, \boxtimes_c$	2
Damgård-Jurik	Class$[n^s]$ / D-Class$[n^s]$	$\boxplus, \boxminus, \boxtimes_c$	$\frac{s+1}{s}$
Length Flexible Damgård-Jurik	Class$[n^s]$ / D-Class$[n^s]$	$\boxplus, \boxminus, \boxtimes_c$	$\frac{s+1}{s}$
Modified Length Damgård-Jurik	Class$[n^s]$ / D-Class$[n^s]$	$\boxplus, \boxminus, \boxtimes_c$	$\frac{s+1}{s}$
Boneh-Goh-Nissim	Subgroup Decision Problem	$\boxplus, \boxminus, \boxtimes_c$ $\boxtimes$ (once)	$\frac{n}{r}$

Table 3.1: A summary of homomorphic cryptosystems.

Chapter 4

Threshold Homomorphic Cryptography

In many situations it is desirable to distribute the decryption process amongst a number of parties such that a message can only be decrypted if a certain qualified subset of these parties participate in the decryption process. For example, consider a cryptographic election protocol where the homomorphic properties of a cryptosystem are used to anonymously calculate the encrypted tally for each candidate. Granting knowledge of the decryption key to any single election official would allow that official to decrypt any ballot in the system, thus learning how an individual voter cast their vote. In order to combat this problem, the decryption key could distributed amongst several election officials, allowing a message to be decrypted if and only if some large enough subset of election officials agree to do so. A cryptosystem that supports such a process is called a *threshold cryptosystem.*

Definition 4.0.1. *A public-key cryptosystem is said to be a* (t, l)*-threshold cryptosystem if a dealer can create an instance of the cryptosystem and distribute* l *secret shares amongst* l *parties such that any subset of* t *or more parties can co-operate to decrypt a message, while any subset of fewer than* t *players can learn nothing about an encrypted message in polynomial time.*

A more general problem than distributing the decryption process is the problem of distributing a single secret value amongst a set of participants such that only a qualified subset of participants can reconstruct the secret. Shamir [92] proposed one of the first threshold secret sharing schemes, based on polynomial interpolation over finite fields, where, as in Definition 4.0.1, a qualified subset of participants is any subset of size t or larger. Shamir's approach forms the basis for many threshold cryptosystems. At the same time, Blakley [12] independently proposed a different secret sharing scheme based on the intersection of l-dimensional hyperplanes; however, in general Shamir's system is more efficient. Shamir's scheme is based on the fact that given t distinct points in a field, there is a unique polynomial of degree $t-1$ that passes through those points, but given $k < t$ points, there are many polynomials of degree t that pass through those points. Blakley's scheme is based off the similar idea that t distinct t-dimensional hyperplanes intersect at a unique point, but $k < t$ hyperplanes intersect at many points.

Cryptosystem 4.0.2. *The Shamir Secret Sharing Scheme*

Let D denote the dealer, let $P = \{P_1, P_2, \ldots, P_l\}$ be the set of participants, and let $t < l$ be the number of players required to reconstruct the secret.

- *Initialization: Let $p > l+1$ be a prime. Then D chooses distinct labels $x_i \in \mathbb{Z}_p^*$ for each participant P_i and makes each x_i public.*
- *Share Distribution: Given a secret $s \in \mathbb{Z}_p$, the dealer uniformly chooses $t-1$ random values from $\mathbb{Z}_p$, denoted $a_1, \ldots, a_{t-1}$, and constructs the polynomial $a(x) = s + \sum_{i=1}^{t-1} a_i x^i$. The value $a(x_i)$ is sent to P_i as that participant's share of the secret.*
- *Share Reconstruction: Let $P_{i_1}, \ldots, P_{i_t}$ be a subset of t participants wishing to reconstruct the secret. The participants collectively use Lagrange interpolation to compute*

$$a(x) = \sum_{j=1}^{t} \left(y_j \prod_{\substack{k=1 \\ k \neq j}}^{t} \frac{x - x_{i_k}}{x_{i_j} - x_{i_k}} \right) \mod p$$

  and recover $s = a(0)$.

If an adversary manages to learn up to $t-1$ shares, then the adversary gains no information about the secret s, as for any possible value of the missing shares, there is a polynomial that passes through these points that also passes through s. Due to the fact that $a(x)$ is a random polynomial, each of these polynomials is possible with equal probability.

Although Shamir secret sharing could be used to distribute the private key of a public key cryptosystem, such a straightforward application is not an ideal method of building a threshold cryptosystem. Once the private key has been reconstructed, any participant has knowledge of the private key and can decrypt any future messages without relying on others. A trivial workaround is to require that each participant submit their secret share to a trusted third party, who then reconstructs the private key, decrypts the message, and sends the result back to the participants. Instead of relying on a trusted party for decryption, it is possible to use Shamir secret sharing as a part of a more complicated protocol that allows the participants to cooperate to decrypt a message, without revealing the private key.

An additional tool is also useful when constructing threshold cryptosystems. If an active adversary has corrupted one or more of the participants, then the adversary is free to send maliciously crafted messages to other participants when combining shares. Thus, during the share combination phase, each player can be required to post not only their share of the secret, but also a proof that shows that their share is the same one received by the dealer. This proof is done in *zero-knowledge*, such that no other participant can deduce any knowledge of the secret share, except that it is valid.

Definition 4.0.3. *A zero-knowledge proof is a protocol that allows one party (the Prover) to convince another party (the Verifier) that a statement is true, such that the following properties are satisfied:*

- *Completeness: If the statement is true, then an honest verifier will accept the statement as true.*
- *Soundness: If the statement is false, then an honest verifier will not accept the statement as true, except with negligible probability.*
- *Zero-Knowledge: Even if the Verifier cheats, the Verifier can only conclude that the Prover's statement is true or false, and he does not learn any additional information.*

If the protocol requires multiple messages to be exchanged by the Prover and Verifier, then the proof is said to be interactive. If the protocol requires only a single message from the Prover, then the protocol is said to be non-interactive.

If each participant proves in zero-knowledge that their posted share is the same share received by the dealer, then the participants can conclude that the reconstructed secret is, in fact, the original secret. A similar notion is the idea of *verifiable secret sharing*, introduced by Chor, Goldwasser, Micali, and Awerbuch [34], where the dealer potentially acts maliciously. In the verifiable setting, even if the dealer is malicious, if the protocol completes then there exists a well-defined secret that the players are able to reconstruct.

4.1 Threshold RSA Signatures

Although not the first threshold RSA variant, Shoup [95] has proposed a threshold version of the RSA signature scheme with several desirable properties:

- Robust and unforgeable (defined later in this section);
- non-interactive share generation and verification;
- the size of individual shares does not grow with the number of participants;
- has a full security proof.

Shoup's threshold variant is also of interest because it essentially computes a modular exponentiation, where the exponent is shared using Shamir secret sharing, without revealing the value of the exponent. This allows Shoup's variant to be used as a general tool for performing "secret modular exponentiations" among a set of participants.

Recall Cryptosystem 1.3.3, the RSA cryptosystem. If $n = pq$ is the product of two distinct primes, then the public encryption key e and the private decryption key d are selected such that $ed \equiv 1 \mod \phi(n)$. Encrypting a message $m \in \mathbb{Z}_n$ is accomplished by calculating $c = m^e \mod n$, and recovering the message is accomplished by calculating $m = c^d \mod n$. Because e and d are modular inverses of each other, a message that has been "encrypted" with the private key d, can be "decrypted" with the public key e. Thus, if Bob wishes to prove he is the author of some message, he can use his private key to encrypt the message, obtaining a signature, and posts this signature alongside the message when he sends it to Alice. To verify that Bob wrote the message, Alice uses Bob's public key to decrypt the signature, and verifies that two messages are equal. In practice, Bob would not sign the entire message m, but rather $H(m)$, where H is a publicly known hash function.

In general, a signature scheme should be *unforgeable*, meaning that an adversary should not be able to create a valid signature for a message, either random or chosen, without knowledge of the private key. Note that simple RSA signature scheme without hashing, presented earlier, is not unforgeable. An adversary can choose an arbitrary signature σ and calculate a message for which σ is a valid signature by $m = \sigma^e \mod n$. Additionally, if the generation or verification of a signature is distributed amongst a group, then the protocol should be *robust* as well, meaning that participants under control of the adversary should not hinder uncorrupted participants from completing the protocol.

Cryptosystem 4.1.1. *Shoup's (t, l)-Threshold RSA Signature Scheme*

Let D denote the dealer, let $P = \{P_1, P_2, \dots, P_l\}$ be the set of participants, and let $t < l$ be the number of players required to reconstruct the secret.

- *Initialization: D chooses distinct primes p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are both primes, and sets $n = pq$ and $m = p'q'$. D then chooses the public exponent $e > l$ and calculates d such that $de \equiv 1 \mod m$. The public label for participant P_i is i. Next, D sets $a_0 = d$, sets a_i to a uniformly random value in $\{0, \dots, m-1\}$ for $1 \leq i < t$, and $f(x) = \sum_{i=0}^{t-1} a_i x^i$. The value $s_i = f(i) \mod m$ is sent to P_i as that participant's share of the secret. In addition to secret shares, the dealer chooses a random square $v \in \mathbb{Z}_n^*$ and makes v along with $v_i = v^{s_i}$ public for $1 \leq i \leq n$.*

- *Signature Share Generation: Let M be the message, let $x = H(M) \in \mathbb{Z}_n^*$ for a hash function H, and let $\Delta = l!$. Participant P_i posts the value $x_i = x^{2\Delta s_i} \mod n$ as their share of the signature, along with a non-interactive zero-knowledge proof that $\log_{x^{4\Delta}} x_i^2 = \log_v v_i$.*

- *Signature Construction: Let $S = \{i_1, \dots, i_t\}$ be a subset of participants of size k wishing to generate the signature. Set*
$$\lambda_{i,j} = \Delta \prod_{j' \notin S - \{j\}} \frac{i - j'}{j - j'},$$
then, if each participant in S posts $x_{i_j} = x^{2\Delta s_{i_j}}$, along with the appropriate zero-knowledge proof, then each participant can verify the proofs and, if all are correct, calculate
$$w = x_{i_1}^{2\lambda_{0,i_1}} \cdots x_{i_t}^{2\lambda_{0,i_t}}.$$
Thus, each participant can compute $w^d = x^{4\Delta^2 d}$, and, using the extended Euclidean algorithm to calculate integers a and b such that $da + 4\Delta^2 b = 1$, can compute $w^b x^a = x^d \mod n$ as a signature on x.

Note that Shoup defines e, d modulo m, rather than $\phi(n) = 4m$. This does not affect the correctness of the scheme, as $\mathbb{Z}_n^* \cong \mathbb{Z}_{\phi(n)} \cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_m$, and $Q_n \cong \mathbb{Z}_m$, where Q_n is the subgroup of squares of $\mathbb{Z}_n^*$. Thus, choosing e, d modulo m simply restricts e, d to be elements of Q_n, which is a proper subgroup of $\mathbb{Z}_n^*$.

Shoup's threshold signature scheme uses Shamir secret sharing as a building block, but has extended it such that the participants are able to collectively compute $x^e \mod n$ instead of e itself. During the initialization phase, shares are created as points on a polynomial, and during reconstruction, they are interpolated in the exponent using Lagrange interpolation. Additionally, during the initialization phase, each participant is issued a verification key $v_i = v^{s_i}$, which is made public. This verification key allows for the construction of a non-interactive zero-knowledge proof that a posted signature share is valid. During the signature construction phase, each participant in S posts $x_{i_j} = x^{2\Delta s_{i_j}}$. Note that $\log_{x^{4\Delta}} x_{i_j}^2 = \log_{x^{4\Delta}} x^{4\Delta s_{i_j}} = s_{i_j}$, and similarly, $\log_v v_{i_j} = \log_v v^{s_{i_j}}$. Thus, if a participant can show that $\log_{x^{4\Delta}} x_i^2 = \log_v v_i$, then the other participants can conclude that the posted value of x_{i_j} is correct.

Let $L(n)$ be the bit-length of n, let H' be a hash function that outputs an L_1-bit integer, and let $Q_n \subseteq \mathbb{Z}_n^*$ be the subset of squares of $\mathbb{Z}_n^*$. A non-interactive protocol can be built using H' that lets the Prover (i.e. a given participant P_{i_j}) demonstrate to a Verifier (i.e. any other participant) that the posted signature share x_{i_j} has been constructed correctly. The Prover first chooses a random integer $r \in \{0, 1, \dots, 2^{L(n)+L_1} - 1\}$ and computes the following values:
$$\begin{aligned} v' &= v^r \\ x' &= x^{4\Delta r} \\ c &= H'(v, x^{4\Delta}, x_{i_j}^2, v', x') \\ z &= s_{i_j} c + r. \end{aligned}$$
The non-interactive proof is the tuple (z, c), from which the Verifier can check that
$$c = H'(v, x^{4\Delta}, x_{i_j}^2, v^z v_{i_j}^{-c}, x^{4\Delta z} x_{i_j}^{-2c}).$$

If the two values are equal, the Verifier accepts the proof. Note that

$$\begin{aligned} v^z v_{i_j}^{-c} &= v^{s_{i_j}c+r} v^{-cs_{i_j}} \\ &= v^r \\ &= v' \end{aligned}$$

and

$$\begin{aligned} x^{4\Delta z} x_{i_j}^{-2c} &= x^{4c\Delta s_{i_j}+4\Delta r} x^{-4c\Delta s_{i_j}} \\ &= x^{4\Delta r} \\ &= x'. \end{aligned}$$

Thus, a Verifier can verify the proof using only publicly available information and the random value z. If H' is a cryptographically secure hash function, then an adversary cannot fake a proof. This zero-knowledge proof is based on a proof given by Chaum and Pedersen [27]; however it has been modified from its original interactive version by using the hash function H' instead of random challenges.

The correctness of signature construction follows from the fact that

$$\Delta f(i) \equiv \sum_{j \in S} \lambda_{i,j} f(j) \mod m.$$

During reconstruction, w is calculated as

$$\begin{aligned} w &= x_{i_1}^{2\lambda_{0,i_1}} \cdots x_{i_k}^{2\lambda_{0,i_t}} \\ &= x^{4\Delta s_{i_1}\lambda_{0,i_1}} \cdots x^{4\Delta^2 s_{i_t}} \\ &= x^{4\Delta(\lambda_{0,i_1}s_{i_1}+\ldots+\lambda_{0,i_t}s_{i_t})} \\ &= x^{4\Delta^2 f(0)} \\ &= x^{4\Delta^2 d}. \end{aligned}$$

Shoup provides a complete proof of security in [96], which is summarized in the following theorem.

Theorem 4.1.2. *([96], Theorem 1) In the random oracle model, for H', the Shamir (t, l)-threshold signature scheme is secure (robust and unforgeable), assuming the standard RSA signature scheme is secure.*

The proof is not presented here; however, a proof for a similar scheme, based on Shoup's proof, is provided in Section 4.6.

Many of the operations in Shoup's scheme are performed in Q_n, the subgroup of squares. Because a participant cannot be sure that a received message has not been maliciously constructed, received messages are squared for a second time to be certain the resulting value is an element of Q_n. The factor Δ is introduced to avoid the computation of inverses during interpolation. When calculating $\lambda_{i,j}$, $(j - j')$ is always a factor of $\Delta = l!$.

4.2 Proving the Semantic Security of a Threshold Cryptosystem

Fouque, Poupard, and Stern [48] propose the following game to prove the semantic security of a threshold cryptosystem:

1. The adversary selects $t-1$ participants to corrupt. The adversary controls the actions of these participants during the decryption steps.

2. The dealer runs the share initialization step and sends each participant their secret share, then makes the public parameters of the cryptosystem public.

3. The adversary chooses a plaintext message M and queries a partial decryption oracle who generates a random encryption of M, and then response with l valid decryption shares (one for each participant), along with the corresponding proofs of correctness. The adversary may repeat this step a polynomially-bounded number of times.

4. The adversary chooses two messages M_0 and M_1 and sends them to an encryption oracle, who randomly chooses a bit b and replies with an encryption of M_B.

5. The adversary may repeat Step 3 a polynomially-bounded number of times.

6. The adversary responds with a bit b', and wins the game if $b' = b$.

A threshold cryptosystem is said to be semantically secure if an adversary can only win the game with probability negligibly greater than $\frac{1}{2}$.

Let $\mathcal{C}$ be a semantically secure cryptosystem, let $\mathcal{C}_t$ be the threshold version of $\mathcal{C}$, and assume, for the sake of contradiction, that there exists a polynomial time adversary $\mathcal{A}$ that can break the semantic security of $\mathcal{C}_t$. By construction, there does not exists a polynomial time adversary that can break the semantic security of $\mathcal{C}$ (i.e., win the IND-CCA1 game with non-negligible advantage); therefore, if an attacker is able to utilize access to $\mathcal{A}$ to break the semantic security of $\mathcal{C}$, the assumption that $\mathcal{C}$ is semantically secure has been contradicted. Thus, the assumption that $\mathcal{A}$ exists must be false.

Recall the IND-CCA1 game (Definition 1.3.6). After being given access to a decryption oracle, the attacker chooses two messages, M_0 and M_1, and receives an encryption of M_b, from which the value of b must be determined. In order to utilize $\mathcal{A}$ to win the IND-CCA1 game, the attacker must provide $\mathcal{A}$ with appropriate inputs (public parameters and a subset of compromised keys) during step 2. During steps 3 and 5, the attacker acts as the partial decryption oracle, replying to any requests issued by $\mathcal{A}$. During step 4, $\mathcal{A}$ outputs two messages, M_0 and M_1, which the attacker submits in the IND-CCA1 game, thus learning c as an encryption of

M_b. The attacker forwards c to $\mathcal{A}$, who then responds with b, which the attacker submits as his response in the IND-CCA1 game.

In order to prove that the semantic security of $\mathcal{C}_t$ is implied by the semantic security of $\mathcal{C}$, it is sufficient to show that, given the public parameters of $\mathcal{C}$, the attacker can simulate all the data expected by $\mathcal{A}$ in steps 2, 3, and 5 such that the simulated data is indistinguishable from real data. If the two distributions are indistinguishable, then $\mathcal{A}$ can be used to break the semantic security of $\mathcal{C}$, thus contradicting the fact that $\mathcal{C}$ is semantically secure. Therefore, if all data in steps 2, 3, and 5 are simulatable with indistinguishable distributions, then an adversary that breaks the semantic security of $\mathcal{A}$ does not exist.

4.3 A Threshold Paillier Cryptosystem

Shoup's threshold signature scheme allows a group of participants to generate a signature by collectively raising a message x to a secret exponent d without revealing the value of d. Recall that decryption in the Paillier cryptosystem can be modified through the appropriate choice of parameters to essentially be a modular exponentiation. Fouque, Poupard, and Stern [48] have extended Shoup's threshold signature scheme to create a threshold variant of the Paillier cryptosystem.

Cryptosystem 4.3.1. *The (t,l)-Threshold Paillier Cryptosystem Let p' and q' be two integers such that $p = 2p' + 1$ and $q = 2q' + 1$ are also prime, and such that $m = p'q'$, $n = pq$, and $\gcd(n, \phi(n)) = 1$. Let a, b, β be random elements of $\mathbb{Z}_n^*$ and set $g = (1 + n)^a b^n \mod n^2$. Then $\mathcal{P} = \mathbb{Z}_n$, $\mathcal{C} = \mathbb{Z}_{n^2}^*$, $\mathcal{R} = \mathbb{Z}_n^*$, and $\mathcal{K} = \{(p, q, m, n, a, b, \beta, g)\}$ where $p, q, m, n, a, b, \beta, g$ are as defined. Let D denote the dealer, and $P = \{P_1, \ldots, P_l\}$ be the set of participants.*

- *Gen: Given a security parameter ϵ, Gen(ϵ) chooses two distinct odd integers p' and q', setting $m = p'q'$, such that $p = 2p'+1$ and $q = 2q'+1$ are also prime, and $\gcd(n, \phi(n)) = 1$ where $n = pq$. Gen then chooses a, b, β at random from $\mathbb{Z}_n^*$ and sets $g = (1 + n)^a b^n$. The public key is $(n, g, am\beta \mod n)$, and the private key is βm.*

- *Share Initialization: D creates an instance of the cryptosystem using Gen, and uses Shamir secret sharing to distribute the secret key, βm, to the members of P by setting $a_0 = \beta m$ and choosing a_i at random from $\{0, \ldots, nm-1\}$ for $i = 1, \ldots t-1$ such that $f(x) = \sum_{i=1}^{t-1} a_i x^i$. The share $s_i = f(i) \mod nm$ is sent to P_i. D then chooses a random v from the subgroup of squares of ${\mathbb{Z}_{n^2}}^*$ and makes $v^{\Delta s_i}$ public for $i = 1, \ldots, n$, where $\Delta = l!$.*

- *Enc: Given a message M and a public key pk, Enc(pk, M) chooses a random $r \in \mathcal{R}$ and computes the ciphertext*

$$c = g^M r^n \mod n^2.$$

- *Decryption Share Generation: Each participant P_i participating in decryption calculates and makes public the partial decryption share $c_i = c^{2\Delta s_i} \mod n^2$, along with a zero-knowledge proof demonstrating that $\log_{c^{4\Delta}} c_i = \log_{v^\Delta} v_i$.*

- *Decryption Share Reconstruction: Let $S = \{i_1, \ldots, i_t\}$ be the labels of a set of t participants who have posted decryption shares with valid proofs. The message can be recovered by computing*

$$M = L\left(\prod_{j\in S} c_{i_j}^{2\lambda_{0,j}} \mod n^2\right) \frac{1}{4\Delta^2 am\beta} \mod n$$

where $L(u) = \frac{u-1}{n}$, and

$$\lambda_{i,j} = \Delta \prod_{j'\in S-\{j\}} \frac{i-j'}{j-j'}.$$

In the original Paillier cryptosystem, the generator g is selected such that the function $f(M, y) = g^M y^n \mod n^2$ is a bijection from $\mathbb{Z}_n \times \mathbb{Z}_n^*$ to $\mathbb{Z}_{n^2}^*$, which holds when $\gcd(L(g^\lambda \mod n^2), n) = 1$. In the threshold variant, g is selected of the form $(1+n)^a \times b^n$, which satisfies $\gcd(L(g^\lambda \mod n^2), n) = 1$ due to the fact that

$$\begin{aligned} g^\lambda \mod n^2 &= ((1+n)^a \times b^n)^\lambda \mod n^2 \\ &= (1+n)^{\lambda a} b^{\lambda n} \mod n^2 \\ &= (1+n)^{\lambda a} \mod n^2 \\ &= (1+\lambda an) \mod n^2. \end{aligned}$$

As in Shoup's threshold RSA scheme, the shares are recombined in the exponent, where

$$\begin{aligned} \prod_{j\in S} c_{i_j}^{2\lambda_{0,j}} \mod n^2 &= c^{4\Delta^2 m\beta} \mod n^2 \\ &= (g^M r^n)^{4\Delta^2 m\beta} \mod n^2 \\ &= g^{4\Delta^2 m\beta M} r^{4\Delta^2 m\beta n} \mod n^2 \\ &= g^{4\Delta^2 m\beta M} \mod n^2 \\ &= (1+n)^{4\Delta^2 zm\beta M} b^{4\Delta^2 m\beta Mn} \mod n^2 \\ &= 1 + 4\Delta^2 am\beta Mn \end{aligned}$$

where the y and b terms vanish due to the fact that $\phi(n^2) = n\phi(n) = 4nm$. Because $am\beta$ is part of the public key, the message can be recovered by applying the function $L(u) = \frac{u-1}{n}$ and multiplying by $(4\Delta^2 am\beta)^{-1}$.

Fouque, Poupard, and Stern provide a proof of security for their threshold variant of the Paillier cryptosystem, based on Shoup's proof of security for threshold RSA, which is summarized in the following theorem.

Theorem 4.3.2. *([48], Theorem 1) If the decisional composite residuosity problem is hard (see Section 3.1.4), then the threshold version of the Paillier cryptosystem is secure against active non-adaptive adversaries in the random oracle model.*

The proof is not presented here; however, a full proof for a similar threshold cryptosystem, also based on Shoup's threshold RSA scheme, is given in Section 4.6.

4.4 Threshold Paillier/Damgård-Jurik Cryptosystems

In addition to generalizing Paillier's cryptosystem to higher powers of n (see Section 3.2.11), Damgård and Jurik [40] provided a threshold variant of their cryptosystem using the same approach as Shoup, also presented later with Nielson [39]. Their approach is similar to Fouque, Poupard, and Stern's threshold variant of the Paillier cryptosystem, although there are a few key differences; namely, Damgård and Jurik's approach works for any modulus n^s, rather than just n^2, and does not rely on additional random values during key generation..

Cryptosystem 4.4.1. *The (t, l)-threshold Damgård-Jurik Cryptosystem*

Assume $t < \frac{l}{2}$. Let p' and q' be two integers such that $p = 2p' + 1$ and $q = 2q' + 1$ are also prime, and set $m = p'q'$ and $n = pq$. Let $s > 0$ such that the plaintext space is $\mathcal{P} = \mathbb{Z}_{n^s}$, the ciphertext space is $\mathbb{Z}_{n^{s+1}}$, and $\mathcal{R} = \mathbb{Z}_{n^{s+1}}$. Let d be an integer such that $d \equiv 0 \mod m$ and $d \equiv 1 \mod n^s$, and set $g = 1 + n$. Then $\mathcal{K} = \{n, s, d, g\}$ where n, s, d, and g are as defined. Let D denote the dealer, and $P = \{P_1, \ldots, P_l\}$ be the set of participants.

- *Gen: Given a security parameter ϵ, and additional parameter $s > 0$, Gen(ϵ, s) chooses two distinct odd integers p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are also prime, and sets $m = p'q'$ and $n = pq$. Gen(ϵ, s) then chooses d such that $d \equiv 0 \mod m$ and $d \equiv 1 \mod n^s$, and sets $g = 1 + n$. The public key is (n, g, s) and the private key is d.*

- *Share Initialization: D creates an instance of the cryptosystem using Gen, and uses Shamir secret sharing to distribute the secret key d by setting $a_0 = d$, choosing at random $a_i \in \{0, \ldots, n^s m - 1\}$ for $i = 1, \ldots t - 1$, and setting $f(x) = \sum_{i=0}^{t-1} a_i x^i \mod n^s m$. The share $s_i = f(i) \mod n^s m$ is sent to P_i as that player's share. D then chooses a random v from the subgroup of squares of ${\mathbb{Z}_{n^{s+1}}}^*$ and makes v public along with $v^{\Delta s_i}$ for $i = 1, \ldots, l$, where $\Delta = l!$.*

- *Enc: Given a message M and a public key pk, Enc(pk, M) chooses a random $r \in \mathcal{R}$ and calculates the ciphertext as*

$$c = g^M r^{n^s} \mod n^{s+1}.$$

- *Decryption Share Generation: Each participant P_i participating in decryption calculates and makes public the partial decryption share $c_i = c^{2\Delta s_i} \mod n^{s+1}$, along with a zero-knowledge proof demonstrating that $\log_{c^{4\Delta}} c_i = \log_{v^{\Delta}} v_i$.*

- *Decryption Share Reconstruction: Let $S = \{i_1, \ldots, i_t\}$ be the labels of a set of t participants who have posted decryption shares with valid proofs. The message can be recovered by computing*
$$c' = \prod_{j \in S} c_{i_j}^{2\lambda_{0,j}} \mod n^{s+1}$$
where
$$\lambda_{i,j} = \Delta \prod_{j' \in S - \{j\}} \frac{i - j'}{j - j'}$$
such that $c' = c^{4\Delta^2 d}$. Each participant then applies the algorithm in Section 3.2.11, to learn the message $M' = 4\Delta^2 M$, and calculates the message by computing
$$M = M'(4\Delta^2)^{-1} \mod n^s.$$

As in the threshold Paillier cryptosystem, during share reconstruction the participants jointly compute $c^{4\Delta^2 d}$, where d is the secret value shared among the participants. Because $g = 1 + n$, each participant learns $c^{4\Delta^2 d} = (1+n)^{4\Delta^2 dM}$, and can apply the decryption algorithm of the original Damgård-Jurik cryptosystem to retrieve the message $M' = 4\Delta^2 M$. Finally, because $4\Delta^2$ is a public value, each participant can recover the original message by calculating $M = M'(4\Delta^2)^{-1} \mod n^s$.

The security of the threshold Damgård-Jurik cryptosystem is summarized in the following theorem.

Theorem 4.4.2. *([40], Theorem 2) Assume the random oracle model and a static adversary that corrupts up to $t - 1$ players from the beginning. Then, given any ciphertext, the decryption protocol outputs the correct plaintext, except with negligible probability. Given an oracle that given a ciphertext returns the corresponding plaintext, the adversary's view of the decryption protocol can be efficiently simulated with a statistically indistinguishable distribution.*

The random oracle model is required to simulate the hash function required during the zero-knowledge proof that a posted share is correct. As with the presentation of Shoup's threshold RSA signature scheme and the threshold Paillier cryptosystem, a formal proof of security is not provided here; however, a full proof for a similar system, also based on Shoup's work, is provided in the next section. A full proof for the system was not given in [40], although an overview of a proof is given in [39].

Recall that s is included as part of the public key, although messages encrypted using a smaller value $0 < s' < s$ will still decrypt correctly, as d will still satisfy

$d \equiv 0 \mod m$ and $d \equiv 1 \mod n^{s'}$. Damgård and Jurik note that there exists a method for computing a new temporary d' that will work for $s' > s$; however, it is an interactive process and must be performed once for every value of s used. In order to eliminate this additional protocol, Damgård and Jurik have proposed a threshold variant of their modified length-flexible cryptosystem [42], which was presented as Cryptosystem 3.2.37.

Cryptosystem 4.4.3. *The (t, l)-threshold Modified Length-Flexible Damgård-Jurik Cryptosystem*

Let $p', q', p, q, n, m, g, h, \mathcal{P}, \mathcal{C}, \mathcal{R}, \mathcal{K}$ be as in Cryptosystem 3.2.37, let D denote the dealer, $P = \{P_1, \ldots, P_l\}$ be the set of participants, and assume that $t \geq \frac{l}{2}$.

- *Gen: Given security parameter ϵ, D creates an instance of Cryptosystem 3.2.37 with public key (n, g, h), and private key α.*
- *Share Initialization: D uses Shamir secret sharing to distribute the secret key α by setting $a_0 = \alpha$, choosing at random $a_i \in \mathbb{Z}_m$ for $i = 1, \ldots t-1$, and setting $f(x) = \sum_{i=0}^{t-1} a_i x^i \mod n^s m$. The share $\alpha_i = f(i) \mod m$ is sent to P_i as that player's share. D then makes verification keys $h_i = h^{\Delta \alpha_i}$ public for $i = 1, \ldots, l$, where $\Delta = l!$.*
- *Enc: Given a public key pk and a message M, Enc(pk, M) chooses s such that $M \in \mathbb{Z}_{n^s}$, a random $r \in \mathcal{R}$, and calculates the ciphertext*

$$c = (G, H) = (g^r \mod n, (h^{4\Delta^2 r} \mod n)^{n^s}(1+n)^M \mod n^{s+1}).$$

- *Decryption Share Generation: Given a ciphertext $c = (G, H)$, each participant P_i participating in decryption calculates and makes public the partial decryption share $G_i = G^{2\Delta\alpha_i} \mod n$, along with a zero-knowledge proof demonstrating that $\log_g h_i = \log_{G^{4\Delta}} {G_i}^2$.*
- *Decryption Share Reconstruction: Let $S = \{i_1, \ldots, i_t\}$ be the labels of a set of t participants who have posted decryption shares with valid proofs. The message can be recovered by computing*

$$G' = \prod_{j \in S} G_{i_j}^{2\lambda_{0,j}} \mod n = h^{4\Delta^2 \alpha} \mod n$$

where

$$\lambda_{i,j} = \Delta \prod_{j' \in S - \{j\}} \frac{i - j'}{j - j'}.$$

Each participant can choose s such that $G, H \in \mathbb{Z}_{n^s}$ and then calculate

$$H' = HG'^{-n^s} = (1+n)^M \mod n^{s+1}.$$

The message is then recovered using the algorithm in Section 3.2.11.

Damgård and Jurik also provide a different variant of their modified threshold system that supports some useful zero-knowledge proofs. The modifications center around ensuring that all computation takes place in the subgroup of squares of $\mathbb{Z}_n$. The encryption function is changed such that

$$c = (G, H) = ((-1)^{b_0} g^r \mod n, (-1)^{b_1} (h^{4\Delta^2 r} \mod n)^{n^s} (1+n)^M \mod n^{s+1})$$

where b_0, b_1 are random bits. Decryption becomes

$$H' = H^2 G'^{-2n^s} = (1+n)^{2M} \mod n^{s+1}$$

and the message can be recovered by applying the algorithm in Section 3.2.11 and dividing by 2.

4.5 A Threshold ElGamal Cryptosystem

Cramer, Gennaro, and Schoenmakers [37] have given a construction of a threshold ElGamal cryptosystem based on similar approach to Shoup's threshold RSA signature scheme. The private key is distributed using Shamir secret sharing, and the interactive version of Chaum and Pederson's proof of knowledge of a discrete logarithm [27].

Cryptosystem 4.5.1. *The (t,l)-threshold ElGamal Cryptosystem*

Let $g \in \mathbb{G}$ be a generator of a cyclic group of order q such that the CDH and DDH problems are hard in $\mathbb{G}$. Let $\mathcal{P} = \mathbb{G}$, $\mathcal{C} = \mathbb{G} \times \mathbb{G}$, and $\mathcal{K} = \{(q, g, x, h) : h \equiv g^x \mod q\}$, where q, g are as above. Let D denote the dealer, $P = \{P_1, \ldots, P_l\}$ be the set of participants.

- *Gen: Give security parameter ϵ, D creates an instance of Cryptosystem 3.2.3 with public key $(\mathbb{G}, q, g, h)$, and private key x.*
- *Share Initialization: D uses Shamir secret sharing to distribute the secret key x by setting $a_0 = x$, choosing at random $a_i \in \mathbb{Z}_q$ for $i = 1, \ldots t-1$, and setting $f(y) = \sum_{i=0}^{t-1} a_i y^i \mod q$. The share $x_i = f(i) \mod q$ is sent to P_i as that player's share. D then calculates and makes public the value $h_j = g^{x_j}$ for each participant.*
- *Enc: Given a public key $pk = (\mathbb{G}, q, g, h)$ and a message $m \in \mathcal{P}$, $Enc(pk, m)$ chooses a random $y \in \mathbb{Z}_q$ and returns the ciphertext (c_1, c_2) where*

  $$c_1 = g^y \mod q$$

  and

  $$c_2 = m \cdot h^y \mod q.$$

- *Decryption Share Generation: Given a ciphertext* $c = (c_1, c_2)$*, each participant* P_i *participant in decryption posts*

$$c_j = c_i^{x_i}$$

  and participates in the protocol of Chaum and Pederson [27] with each other participant to verify that each share is constructed correctly; that is, proves in zero-knowledge that $\log_g h_j = \log_x w_j$.

- *Decryption Share Reconstruction: Let* $S = \{i_1, \ldots, i_t\}$ *be the labels of a set of* t *participants who have posted decryption shares with valid proofs. The message is recovered by computing*

$$m = \frac{c_2}{\prod_{i \in S} w_j^{\lambda_{S,j}}}$$

  where

$$\lambda_{S,j} = \prod_{i \in S - \{j\}} \frac{i}{i-j}.$$

The (t, l)-threshold ElGamal cryptosystem is slightly simpler than threshold cryptosystems based on Shoup's approach. The underlying idea of distributing shares using Shamir secret sharing, providing zero-knowledge proofs of that shares are constructed correctly, and performing interpolation in the exponent without revealing the secret are common to both systems, but Shoup's approach is suited to groups whose order is the product of two distinct prime powers p, q of the form $p = 2p' + 1$ and $q = 2q' + 1$ for primes p' and q'. Shoup's approach allows for the construction of simple simulation-based proofs of security, as described in Section 4.2 and demonstrated in Section 4.6. This is due to the fact that Shoup restricts most parameters to the subgroup of squares, allowing for all received values to be squared, thus ensuring that arithmetic is confined to the subgroup of squares. Uniformly random elements of the subgroup of squares can also be simulated by an adversary using only public knowledge. The proof of security given in the next section demonstrates these concepts.

4.6 A New Threshold Boneh-Goh-Nissim Cryptosystem

In the conclusion of [48], Fouque, Poupard, and Stern leave the open problem of creating threshold variants of other Paillier-like cryptosystems. At the time the problem was posed, the Boneh-Goh-Nissim cryptosystem had not yet been discovered. In this section, the first construction of a threshold variant of the BGN cryptosystem is given, with a complete proof of security that builds off the approach of Shoup's original threshold RSA signature scheme, using the security model defined by Fouque, Poupard, and Stern.

The Boneh-Goh-Nissim (BGN) cryptosystem (see Section 3.2.12) is very similar to the Paillier cryptosystem; however, it sacrifices efficient decryption for the ability to perform a single homomorphic multiplication, in addition to the usual homomorphic addition. Because the structure of the BGN cryptosystem is very similar to the Paillier cryptosystem, it is possible to use Shoup's approach to define a threshold variant of BGN in the same way that Shoup's method was used to create threshold variants of the Paillier and Damgård-Jurik cryptosystems.

The threshold version of BGN will follow the same structure as the previously presented threshold systems. Each participant will receive a share of the secret key s_i, and a verification key $v_i = v^{\Delta s_i}$ is made public for each participant. In the previous systems, a non-interactive zero-knowledge proof of knowledge is posted along with each decryption share to demonstrate that the decryption share is constructed appropriately. The additional group structure available in the BGN cryptosystem allows this proof to be greatly simplified. Recall that the BGN cryptosystem is defined over a group with a bilinear pairing e, with the property that $e(x^a, y^b) = e(x, y)^{ab}$ for any two group elements x and y. When a participant posts the decryption share $c_i = c^{2\Delta s_i}$, no additional information is required to prove that c_i is constructed appropriately. Any other participant can verify that $e(v, c_i) = e({v_i}^2, c)$, accepting the decryption share as valid if the two values are equal. This works due to the fact that

$$\begin{aligned} e(v, c_i) &= e(v, c^{2\Delta s_i}) \\ &= e(v, c)^{2\Delta s_i} \\ &= e(v^{2\Delta s_i}, c) \\ &= e({v_i}^2, c). \end{aligned}$$

Thus, verifying shares in the threshold BGN cryptosystem requires less work than verifying shares in the threshold Paillier variants.

Cryptosystem 4.6.1. *The (t, l)-threshold Boneh-Goh-Nissim Cryptosystem*

Let p' and q' be two integers such that $p = 2p'+1$ and $q = 2q'+1$ are also prime, and set $m = p'q'$ and $n = pq$. Let $\mathbb{G}, \mathbb{G}_1$ be two multiplicative groups of order n with a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, and let g be a random generator of $\mathbb{G}$ and h be a random generator of the subgroup of $\mathbb{G}$ of order p. Let $T < q$, then $\mathcal{P} = \mathbb{Z}_T$, $\mathcal{C} = \mathbb{G}$, $\mathcal{R} = \mathbb{Z}_n$, and $\mathcal{K} = \{(n, p, q, T, \mathbb{G}, \mathbb{G}_1, e, g, h, g^p)\}$ where $(n, p, q, T, \mathbb{G}, \mathbb{G}_1, e, g, h)$ are defined as above. Let D denote the dealer, and let $P = \{P_1, \ldots, P_l\}$ be the set of participants.

- *Gen: Given security parameter ϵ, Gen(ϵ) chooses two distinct odd $\frac{\epsilon}{4}$-bit primes p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are also prime, and sets $m = p'q'$ and $n = pq$. Gen(ϵ) then selects a positive integer $T < q$ such that solving discrete logs where the exponent is small than T is tractible. Using the method described in Section 3.1.7, Gen(ϵ) then chooses two multiplicative groups $\mathbb{G}, \mathbb{G}_1$ of order n that support a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$,* *as well as a random generators $g, u \in \mathbb{G}$, and sets $h = u^q$ such that h is a generator of the subgroup of order p. The public key is $(n, g, h, g^p, \mathbb{G}, \mathbb{G}_1, e)$, and the private key is p.*

- *Share Initialization: D creates an instance of the cryptosystem using Gen, and uses Shamir secret sharing to distribute the private key p by setting $a_0 = p$, choosing at random $a_i \in \{0, \ldots, m-1\}$ for $i = 1, \ldots t-1$, and setting $f(x) = \sum_{i=0}^{t-1} a_i x^i \mod m$. The share $s_i = f(i) \mod m$ is sent to P_i as that participant's share. D then chooses a random v from the subgroup of squares of $\mathbb{Z}_n{}^*$ and makes $v^{\Delta s_i}$ public for $i = 1, \ldots, l$, where $\Delta = l!$.*

- *Enc: Given a message $m \in \mathcal{P}$ and a public key pk, Enc(pk, m) chooses a random $r \in \mathcal{R}$ and calculates the ciphertext*

$$c = g^m h^r \in \mathbb{G}.$$

- *Decryption Share Generation: Each participant P_i participating in decryption calculates and makes public the partial decryption share $c_i = c^{2\Delta s_i}$.*

- *Decryption Share Reconstruction: Assume, without loss of generality, that participants $P_1, \ldots, P_t$ have submitted decryption shares. Each participant verifies that $e(v, c_i) = e(v_i{}^2, c)$, aborting the protocol if any share is not valid. The message is recovered by calculating*

$$c' = \prod_{j=1}^{t} c_j^{2\lambda_{0,j}} \in \mathbb{G}$$

where

$$\lambda_{i,j} = \Delta \prod_{\substack{j'=1 \\ j' \neq j}}^{t} \frac{i - j'}{j - j'}.$$

such that $c' = c^{4\Delta^2 p}$. The message is recovered by taking the discrete logarithm of c' to the base $g^{4\Delta p}$.

As described in Section 4.2, proving the semantic security of the threshold BGN cryptosystem will be accomplished by demonstrating that an adversary that can break the semantic security of the threshold BGN cryptosystem can also break the semantic security of the BGN cryptosystem. Thus, the security of the basic BGN cryptosystem implies the security of the threshold variant. The proof itself consists of demonstrating how an attacker can take an instance of the BGN cryptosystem and simulate an instance of the threshold variant in a manner that is indistinguishable from a real instance of the threshold variant. Although the basic and threshold cryptosystems are similar, a problem arises when attempting to simulate the value g^p, required in the threshold version to perform the discrete logarithm when recovering a message.

Given an instance of the BGN cryptosystem, let the *modified BGN cryptosystem* be the same instance of the cryptosystem with the value g^p made public. Given g^p, an attacker that cannot solve discrete logarithms is unable to recover p through a discrete logarithm computation. Recall that during the semantic security game, the attacker is given access to a decryption oracle, to which the ciphertext $c = g^p h^r$ can be queried, thus revealing the value of p. Altering the oracle to not respond to a request that decrypts to p is insufficient to protect against this attack, as the attacker could exploit the malleability of the BGN cryptosystem to query an encryption of $ap + b$ for know values of a and b. In order prevent the oracle from revealing the secret key, the assumption must be made that T, the upper bound on message size, is much smaller than p. Thus, any message returned by the oracle is reduced modulo T, and the attacker may only learn $p \mod T$. This assumption is reasonable in practice, as T will always be chosen to be much smaller than p, due to the fact that decryption requires $O(\sqrt{T})$ time when calculating the discrete logarithm to recover M.

Because the value g^p is necessary for the threshold BGN cryptosystem, its security will be demonstrated with respect to the security of the modified BGN cryptosystem under the threshold semantic security game defined in Section 4.2. With practical parameter selection (i.e. $T << p$), the modified BGN cryptosystem does not appear to grant an adversary any non-negligible advantage in breaking the cryptosystem.

For the duration of this section, let $\mathcal{C}$ denote the modified *BGN* cryptosystem, let $\mathcal{C}_t$ denote the threshold version, and let $\mathcal{A}$ be a polynomial time adversary that, with access to $t-1$ of l secret shares, can win the threshold semantic security game with non-negligible advantage. Fix $n = pq = (2p'+1)(2q'+1)$ and $m = p'q'$ where p', q', p, q are distinct odd primes, and let Q_n denote the subgroup of squares of $\mathbb{Z}_n^*$.

Lemma 4.6.2. *Any subset of $t-1$ shares are independently random elements of $\mathbb{Z}_m$.*

Proof. The result follows from the fact that each $s_i = f(i)$ where f is a random polynomial of degree $t-1$ with coefficients from $\mathbb{Z}_m$. □

Lemma 4.6.3. *The statistical distance between the uniform distribution on $X = \{0, \ldots, \lfloor \frac{n}{4} \rfloor\}$ and the uniform distribution on $Y = \{0, \ldots, m\}$ is $O(n^{-\frac{1}{2}})$; i.e., the advantage an adversary has in distinguishing between a polynomially sized sample from each distribution is negligible in ϵ, the bit-length of n.*

Proof. Given that $Pr[X = a] = \frac{1}{\lfloor \frac{n}{4} \rfloor}$ for $a \in \{0, \ldots, \lfloor \frac{n}{4} \rfloor\}$ and 0 elsewhere, and given that $Pr[Y = b] = \frac{1}{m}$ for $b \in \{0, \ldots, m\}$ and 0 elsewhere, the statistical distance between X and Y is given by

$$\begin{aligned}
\Delta(X,Y) &= \frac{1}{2}\sum_{i\in\mathbb{N}} |Pr[X=i]-Pr[Y=i]| \\
&= \frac{1}{2}\left(m\left|\frac{1}{\lfloor\frac{n}{4}\rfloor}-\frac{1}{m}\right| + (\left\lfloor\frac{n}{4}\right\rfloor - m)\left|\frac{1}{\lfloor\frac{n}{4}\rfloor}\right|\right) \\
&= \frac{1}{2}\left(\left|\frac{m}{\lfloor\frac{n}{4}\rfloor}-1\right| + \left|1-\frac{m}{\lfloor\frac{n}{4}\rfloor}\right|\right) \\
&= 1-\frac{m}{\lfloor\frac{n}{4}\rfloor} \\
&= \frac{\lfloor\frac{n}{4}\rfloor - m}{\lfloor\frac{n}{4}\rfloor} \\
&= \frac{O(n^{\frac{1}{2}})}{O(n)} \\
&= O(n^{-\frac{1}{2}}).
\end{aligned}$$

Thus, an adversary cannot distinguish between uniform distributions on X and Y with more than negligible advantage in the bit-size of n. □

Lemma 4.6.4. *(Shoup [95]) Q_n is a cyclic group of order m.*

Proof. Recall that $\phi(n) = (p-1)(q-1) = 4m$. Then $\mathbb{Z}_n^*$ is isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_m$. Let J_n denote the set of elements of $\mathbb{Z}_n^*$ having Jacobi symbol 1 (see Section 3.1.3), then $Q_n \subset J_n \subset \mathbb{Z}_n^*$. Exactly half the elements of $\mathbb{Z}_n^*$ are in J_n, and exactly half the elements of J_n are in Q_n. Thus Q_n has order m and is isomorphic to $\mathbb{Z}_m$, i.e., it is cyclic. □

The following two equations relating to Lagrange interpolation will also be required:

$$\lambda_{i,j}^S = \Delta \prod_{j'\notin S-\{j\}} \frac{i-j'}{j-j'}, \tag{4.1}$$

and

$$\Delta f(i) \equiv \sum_{j\in S} \lambda_{i,j} f(j) \mod m. \tag{4.2}$$

Note that by including the value Δ when calculating $\lambda_{i,j}^S$, it is unnecessary to compute any inverses, as $(j-j')$ divides $\Delta = l!$ for each value of j'.

Given these preliminary results, it is now possible to show that each of the steps of the threshold semantic security game can be simulated in a manner that is indistinguishable from a real run of $\mathcal{C}_t$, allowing an attacker against $\mathcal{C}$ to break the semantic security of the cryptosystem by utilizing $\mathcal{A}$'s ability to break the semantic security of $\mathcal{C}_t$.

Lemma 4.6.5. *(Indistinguishability of step 2) Given the public key for $\mathcal{C}$, the attacker can provide $\mathcal{A}$ with a simulated tuple (n, $\mathbb{G}$, $\mathbb{G}_1$, e, g, h, g^p, v, $\{v_1, \ldots, v_l\}$, $\{s_1, \ldots, s_t\}$) such that this tuple is indistinguishable from $\mathcal{A}$'s view of a real instance of $\mathcal{C}_t$.*

Proof. The attacker has knowledge of the public key $(n, g, h, \mathbb{G}, \mathbb{G}_1, e)$ of $\mathcal{C}$, and may forward these values to $\mathcal{A}$ unmodified. In the modified BGN cryptosystem, the attacker also has knowledge of g^p.

Without loss of generality, assume that participants $P_1, \ldots, P_{t-1}$ are to be corrupted by $\mathcal{A}$. By Lemma 4.6.2, any subset of $t-1$ shares are independently random elements of $\mathbb{Z}_m$. Thus, the attacker randomly chooses $s_1, \ldots, s_{t-1}$ from the set $\{0, \ldots, \lfloor \frac{n}{4} \rfloor\}$. By Lemma 4.6.3, $\mathcal{A}$ cannot distinguish between the uniform distribution on $\{0, \ldots, \lfloor \frac{n}{4} \rfloor\}$ and the uniform distribution on $\{0, \ldots, m\}$. Thus, the values $s_1, \ldots, s_{t-1}$ may be forwarded to $\mathcal{A}$ and are indistinguishable from secret shares generated by an actual instance of $\mathcal{C}_t$.

To simulate v, the attacker chooses v to be a valid ciphertext encrypting a known message $2m_0$. Let m_0 be a random message, and calculate $v = (g^{m_0}h^r)^2$ as a random valid encryption of $2m_0$. Because m_0 and g^p are known values, the attacker now has

$$\begin{aligned}(g^p)^{2m_0} &= (g^p)^{2m_0}(h^{2r})^p \\ &= (g^{2m_0}h^{2r})^p \\ &= v^p \\ &= v^{f(0)}.\end{aligned}$$

as a known value. Furthermore, v is a generator of the subgroup of squares, except with negligible probability. The verification keys $v_1, \ldots, v_{t-1}$ are calculated as $v_i = v^{\Delta s_i} \mod n$ for $i \in \{1, \ldots, t-1\}$. For $i \in \{t, \ldots, l\}$, Lagrange interpolation can be used to calculate v_i for $i \in \{t, \ldots, l\}$ without knowledge of s_i using $v^p = v^{f(0)}$.

$$\begin{aligned}v_i &= (g^p)^{2m_0\lambda^S_{i,0}} \prod_{j=1}^{t-1} v^{2s_j\lambda^S_{i,j}} \\ &= \prod_{j \in S} v^{2s_j\lambda^S_{i,j}} \\ &\equiv v^{\Delta f(i)} \mod m \\ &\equiv v^{\Delta s_i} \mod m\end{aligned}$$

by Equations 4.1 and 4.2, where S is the set of known values of s_j along with the point at $v^{f(0)}$. Because m_0 was chosen uniformly at random, v is a uniformly random generator of the subgroup of squares, except with negligible probability, and hence, the simulated verification keys v_i are indistinguishable from a real run of $\mathcal{C}_t$.

The attacker constructs the tuple $(n, \mathbb{G}, \mathbb{G}_1, e, h, g^p, v, \{v_1, \ldots, v_l\}, \{s_1, \ldots, s_t\})$, using the values calculated above, and sends it to $\mathcal{A}$, who cannot distinguish it from a real instance of $\mathcal{C}_t$. $\square$

Lemma 4.6.6. *(Indistinguishability of step 3) During the step 3 of the threshold semantic security game, given a message M, the attacker can provide $\mathcal{A}$ with simulated decryption shares of a random encryption of M, $\{c_1, \ldots, c_l\}$, along with zero-knowledge proofs that each c_i is correct such that $\mathcal{A}$ cannot distinguish between the values provided by the attacker and values provided from a real instance of $\mathcal{C}_t$.*

Proof. During step 3, $\mathcal{A}$ provides the attacker with a message M, who must then respond with partial decryption shares for an encryption of M for each participant. The attacker first calculates the ciphertext $c = g^M h^r \in \mathbb{G}$ for a random integer $r \in \mathbb{Z}_n$. The attacker has knowledge of $\{s_1, \ldots, s_{t-1}\}$, and can thus correctly calculate $c_i = c^{2\Delta s_i}$ for $i \in \{1, \ldots, t-1\}$. The attacker can use the public parameter g^p and the message m to compute $c_0 = c^{2\Delta f(0)}$ by calculating

$$\begin{aligned} (g^p)^{2\Delta M} &= (g^M)^{2\Delta p}(h^{2^\Delta r})^p \\ &= (g^M h^r)^{2\Delta p} \\ &= (g^M h^r)^{2\Delta f(0)} \\ &= c_0. \end{aligned}$$

The attacker then uses Lagrange interpolation to calculate the remaining values of c_i for $i \in \{t, \ldots, l\}$

$$\begin{aligned} c_i &= c_0^{\lambda^S_{i,0}} \prod_{j=1}^{t-1} c^{2s_j \lambda^S_{i,j}} \\ &= \prod_{j \in S} c^{2s_j \lambda^S_{i,j}} \\ &\equiv c^{2\Delta f(i)} \mod m \\ &\equiv c^{2\Delta s_i} \mod m \end{aligned}$$

where S is the set of known values of s_j, along with the point at c_0. Thus, the attacker can provide $\mathcal{A}$ with valid decryption shares $c_0, \ldots, c_l$ that are indistinguishable from an actual instance of $\mathcal{C}_t$.

Providing proofs of validity for each decryption share is unnecessary, as the pairing e provides the ability to verify decryption shares automatically. This is due to the fact that

$$e(v, c_i) = e(v^2, c) = e(v, c)^{2\Delta s_i}$$

for properly constructed shares. Thus, each share is valid if and only if $e(v, c_i) = e(v^2, c)$.

The attacker returns the tuple $(c_1, \ldots, c_l)$ to $\mathcal{A}$, who is unable to distinguish it from an actual instance of $\mathcal{C}_t$. $\square$

Lemma 4.6.7. *(Indistinguishability of step 5) During the step 5 of the threshold semantic security game, given a message M, the attacker can provide $\mathcal{A}$ with simulated decryption shares $\{c_1, \ldots, c_l\}$ and zero-knowledge proofs that each c_i is correct such that $\mathcal{A}$ cannot distinguish between the values provided by the attacker and values provided from a real instance of $\mathcal{C}_t$.*

Proof. Step 5 is identical to step 3. □

Utilizing these lemmas, the security of the threshold BGN cryptosystem can now be summarized.

Theorem 4.6.8. *The (t, l)-threshold BGN cryptosystem is semantically secure if the modified BGN cryptosystem is semantically secure.*

Proof. As established in Section 4.2, the threshold BGN cryptosystem is semantically secure if given an instance of the modified BGN cryptosystem, an attacker can simulate steps 2, 3, and 5 such that the simulated values are indistinguishable from an actual instance of the threshold BGN cryptosystem. Lemmas 4.6.5, 4.6.6, and 4.6.7 establish the indistinguishability of each step, and thus the (t, l)-threshold BGN cryptosystem is semantically secure. □

The problem of creating a threshold variant of the BGN cryptosystem that does not require the public parameter g^p is still an open problem. Such an approach would have to include a distributed method of calculating a discrete logarithm where the base is shared among the participants, or would require the modification of the cryptosystem to allow the discrete logarithm to rely only on information that is public in the basic BGN cryptosystem.

4.7 On the (Im)possibility of Other Homomorphic Threshold Cryptosystems

As mentioned in Section 4.6, in the conclusion of [48], Fouque, Poupard, and Stern leave the open problem of creating threshold variants of other Paillier-like cryptosystems, such as the Goldwasser-Micali cryptosystem (Section 3.2.2), the Benaloh cryptosystem (Section 3.2.4), the Naccache-Stern cryptosystem (Section 3.2.5), and the Okamoto-Uchiyama cryptosystem (Section 3.2.7). This section addresses the existence of threshold variants of these systems using Shoup's approach, concluding that Shoup's method cannot be utilized to create new threshold variants.

Shoup's technique of distributing shares of a secret key and performing interpolation of the shares in the exponent is very flexible, and has allowed for simple threshold variants of the Paillier and Damgård-Jurik cryptosystems. In order to use Shoup's technique for the BGN cryptosystem, the basic cryptosystem was modified to make g^p a public parameter, where p is the secret shared among the participants.

The BGN cryptosystem differs from the Paillier cryptosystem in that the private key is required not only for exponentiation, but also in the calculation of a discrete logarithm. Thus, g^p was necessarily made a public parameter.

Recall that decryption in the Goldwasser-Micali cryptosystem requires determining if the ciphertext c is a quadratic residue modulo $n = pq$, which is tested by computing

$$c^{\frac{p-1}{2}} \mod p.$$

Decryption in the Okamoto-Uchiyama cryptosystem requires the computation of

$$\frac{L(c^{p-1} \mod p^2)}{L(g^{p-1} \mod p^2)} \mod p$$

where p is the private key, and decryption in the Naccache-Stern cryptosystem requires Chinese remaindering modulo secret values p_i. Each of these cryptosystems differs from the Paillier, Damgård-Jurik, and BGN cryptosystems in that decryption requires a reduction modulo a secret value; i.e., the private key is used for more than just exponentiation.

Theorem 4.7.1. *Let $n = pq$ and assume there exists a protocol $\mathcal{P}$ that allows a set of participants to collectively compute $a \mod p$ for a known value a and secret value p. Then $\mathcal{P}$ may reveal the value of p.*

Proof. Let $a_p = a \mod p$ be the result returned by $\mathcal{P}$. Then p divides $a - a_p$ (i.e., $a - a_p = kp$ for some k), and $\gcd(a - a_p, n)$ is either n or p. Thus, if $\mathcal{P}$ implements reduction modulo p, then $\mathcal{P}$ may leak information about the secret value p. □

This result demonstrates that Shoup's approach cannot be utilized in conjunction with a secondary protocol $\mathcal{P}$ that reduces a known value modulo a secret value. Because such a reduction is necessary in the Goldwasser-Micali, Naccache-Stern, and Okamoto-Uchiyama cryptosystems, Shoup's method of interpolation in the exponent cannot be utilized in conjunction with a secondary protocol to reduce modulo p. Thus, if secure threshold variants of these cryptosystems exist, an entirely new approach that combines exponentiation and reduction modulo a secret value must be devised, or the cryptosystems must be altered to eliminate the need to perform reductions modulo a secret value. For example, decryption in the Goldwasser-Micali cryptosystem requires the computation of $c^{\frac{p-1}{2}} \mod p$. Although computing this value using Shoup's approach is not possible, there may exists a protocol which can determine if $c^{\frac{p-1}{2}} \equiv 1 \mod p$ without revealing the value of p. Such a protocol could be used to build a threshold variant of the Goldwasser-Micali cryptosystem.

The creation of a threshold variant of the Benaloh cryptosystem remains an open problem. Cramer, Gennaro, and Schoenmakers [37] have outlined a Benaloh-like cryptosystem and noted that decryption requires computing $c^{\frac{(p-1)(q-1)}{q}} \mod n$ for several values of q until $c^{\frac{(p-1)(q-1)}{q}} \equiv 1 \mod n$. They note that this is essentially the same as RSA decryption, and thus, any approach to distributed RSA decryption, such as the approaches by Gennaro, Jarecki, Krawczyk, and Rabin [56, 58], could be utilized to decrypt Benaloh ciphertexts. This approach does not address the fact that group decryption must be performed several times to decrypt a ciphertext, which makes similar threshold cryptosystems, such as the Paillier and BGN cryptosystem, much more appealing in practice. An approach that requires only a single joint decryption is still an interesting open problem.

Chapter 5

Applications of Homomorphic Cryptography

The ability to perform simple deterministic computations on encrypted data make homomorphic cryptosystems ideal for creating privacy preserving protocols. In this section a brief overview is provided to demonstrate how homomorphic cryptography is utilized in some of these protocols. In general, the protocols presented are meant to be general building blocks for further applications. For example, utilizing homomorphic cryptography to perform simple set operations provides tools that can be used to construct even more complicated protocols, such as complicated database queries. In the case of strong conditional oblivious transfer, presented in Section 5.3, a new construction of the protocol using the BGN cryptosystem is provided, making the protocol secure against a malicious receiver.

As a more specific example of how these basic protocols may be extended, Zhong [105], and Zhong, Goldberg and Hengartner [106], have applied a variety of the protocols given in this section to build new location privacy protocols. These approaches include using the homomorphic properties of a cryptosystem to allow two parties to calculate the distance between them without revealing their actual locations, and the greater than strong condition oblivious transfer protocol presented in Section 5.3 is used to obliviously transfer location based information depending on whether or not this distance is less than some specified threshold. The set operations presented in Section 5.1.2 allow two parties to submit a set of locations and to determine whether or not the sets are disjoint. Based on the result, they can choose which location specific information they with to share.

In general, privacy preserving protocols model a situation in which Bob has a secret function f, and Alice has a set of inputs, $x_1, \ldots, x_k$, for which she wishes to learn $f(x_1, \ldots, x_k)$, without revealing her inputs. If Bob's function can be modeled solely using the operations provided by a homomorphic cryptosystem, then Alice can submit encrypted inputs to Bob, who then performs the necessary homomorphic operations, randomizes the resulting ciphertext, and sends the encrypted result back to Alice. Upon decryption, Alice learns $y = f(x_1, \ldots, x_k)$, but she can deduce no information about f that is not revealed by y itself. Similarly, a group of participants may wish to collectively compute the result of some public function, but without individually revealing their inputs. This situation models a voting system, where each participant has a vote, and wishes to learn the final tally without revealing who they voted for.

Privacy preserving protocols utilizing homomorphic cryptography are usually considered secure in one of two models: The honest-but-curious model, or the malicious model. The security of these two models is defined with respect to an ideal implementation of a protocol where all participants securely transmit their inputs to a trusted third party, who then performs the required computation and returns the result. The information an adversary can gain under the honest-but-curious or malicious models should be identical to what the adversary can gain with respect to the ideal implementation.

In the *honest-but-curious* (HBC) model, all participants honestly provide proper inputs at each step of the protocol, and properly perform any calculations expected of them. The model is named "honest-but-curious" because each participant is honest in the sense that they do not provide false input, but curious in the sense that they will attempt to gain extra information if the protocol makes it possible, including colluding with other participants. Thus, the assumption that fewer than t of n participants collude, for some pre-specified value of t, is often required. A protocol is secure in the HBC model if the amount of information gained by each participant is identical to the information gained when utilizing a trusted third party.

Unlike HBC participants, malicious participants may attempt to deviate from the protocol or provide invalid inputs. Thus, a malicious participant has the ability to completely disrupt the protocol and prevent a correct result from ever being computed. Because controlling the behaviour of a malicious participant is impossible, security in the malicious setting is limited to preventing the malicious party from learning anything about another participant's input, aside from what the result of the protocol implies. As stated earlier, this is formalized by requiring that the amount of information a malicious participant may gain in a run of the protocol is identical to the amount of information gained in an ideal implementation using a trusted third party, where each participant securely transmits each input to the trusted party, who then computes and returns the result. A malicious participant learns only what is implied by the result of the computation, and nothing else.

It is important to note that a protocol being secure in the malicious model does not necessarily mean that a malicious participant cannot compromise privacy. For example, in calculating a set union for two participants, a malicious participant could submit the empty set, thus learning the other participant's complete set from the result. Because the same strategy is possible using a trusted third party, this is not considered a failure of the protocol.

5.1 Manipulating Encrypted Polynomials

Most additively homomorphic cryptosystems, such as the Paillier cryptosystem, lend themselves naturally to computations on encrypted polynomials. Let f, g be two polynomials of degree less than or equal to k, where $f = a_0 + a_1x + \ldots + a_kx^k$ and $g = b_0 + b_1x + \ldots + b_kx^k$. Adopting the shorthand notation $E(x)$ to represent an encryption of x, let $E(f) = \{E(a_0), E(a_1), \ldots, E(a_k)\}$ and $E(g) = \{E(b_0), E(b_1), \ldots, E(b_k)\}$. If E represents the encryption function from an additively homomorphic cryptosystem that allows for two encrypted messages to be summed (denoted $\boxplus$), along with addition and multiplication by a constant (denoted $\boxtimes$), the following operations are possible on f and g:

- Encrypted sum of encrypted polynomials: Given $E(f)$ and $E(g)$, let $f + g = c_0 + c_1x + \ldots + c_kx^k$. Then

  $$E(f+g) = \{E(c_0), E(c_1), \ldots, E(c_k)\},$$

  where

  $$E(c_i) = E(a_i) \boxplus E(b_i).$$

- Encrypted product of an encrypted polynomial and a known polynomial: Given $E(f)$ and g, let $fg = c_0 + c_1x + \ldots + c_{2k}x^{2k}$. Then

  $$E(fg) = \{E(c_0), E(c_1), \ldots, E(c_{2k})\},$$

  where

  $$E(c_l) = \sum_{\substack{i+j=l \\ 0\le i\le k \\ 0\le j\le k}} E(a_i) \boxtimes b_j$$

  and the summation is performed using $\boxplus$.

- Multiplication by a constant: Given $E(f)$, let $cf = c_0 + c_1x + \ldots c_kx^k$ for a constant c. Then

  $$E(cf) = \{E(c_0), E(c_1), \ldots, E(c_k)\},$$

  where

  $$E(c_i) = E(a_i) \boxtimes c_i.$$

- Encrypted evaluation of an encrypted polynomial at a known point: Given $E(f)$ and a point x

  $$E(f(x)) = \sum_{i=0}^{k} E(a_0) \boxtimes x^i$$

  where the summation is performed using $\boxplus$.

- Encrypted formal derivative: Given $E(f)$, let $f' = c_0 + c_1x + \ldots, c_{k-1}x^{k-1}$. Then
$$E(f') = \{E(c_0), E(c_1), \ldots, E(c_{k-1})\},$$
where
$$E(c_i) = E(a_{i+1}) \boxtimes (i+1).$$
- Encrypted integral: Given $E(f)$, let $\int f = c_1x + \ldots + c_{k+1}x^{k+1}$. Then
$$E(\int f) = \{E(0), E(c_1), E(c_2), \ldots, E(c_{k+1})\},$$
where
$$E(c_i) = E(a_{i-1} \boxtimes i^{-1}).$$

After performing any of these operations it is important to re-randomize the resulting ciphertext(s) to ensure that the sequence of operations performed cannot be recovered by an adversary with knowledge of the random values used during the initial encryption of each message.

5.1.1 Multivariate Polynomials and 2-DNF Formulas

Additively homomorphic cryptosystems, such as the Paillier cryptosystem, naturally support several common operations on encrypted polynomials. The fact that operations on ciphertexts are limited to addition restricts additively homomorphic cryptosystems to single-variable polynomials. The BGN cryptosystem, however, has the additional property that a single multiplication can be performed on encrypted messages, allowing it to naturally extend to operations on polynomials whose individual terms have total degree of 2.

Decrypting a BGN ciphertext requires the solution of a discrete logarithm. Thus, in practice, messages must be bounded by some small value T. This introduces practical limitations on the use of the BGN cryptosystem, but makes it ideal for operations performed on binary messages. If the plaintext space is $\mathcal{P} = \{0, 1\}$, then the discrete logarithm computation is greatly simplified. Let $c = g^m h^r$ be an encryption of 0 or 1. Then m can be recovered by first computing
$$c' = c^p = (g^p)^m,$$
which allows m to be recovered by
$$m = \begin{cases} 0 \text{ if } c' = 1 \\ 1 \text{ if } c' = g^p. \end{cases}$$

As noted by Boneh, Goh, and Nissim [14], in this setting the BGN cryptosystem naturally models and allows for the encrypted evaluation of 2-DNF formulas.

Definition 5.1.1. *A statement in boolean logic is said to be in disjunctive normal form (DNF) if it is expressed as a disjunction of one or more conjunctions. A statement is in second disjunctive normal form (2-DNF) if each conjunction consists of only two variables, i.e.*

$$(x_{i_1} \wedge x_{i_2}) \vee (x_{i_3} \wedge x_{i_4}) \vee (x_{i_5} \wedge x_{i_6}) \vee \ldots .$$

In order to model a 2-DNF formula as a multivariate polynomial, it must be "arithmetized" by replacing each instance of $\wedge$ with $\cdot$, and each instance of $\vee$ with $+$. Thus, the $\wedge$ operation is replaced by multiplication, such that $x_1 x_2 = 1$ if and only if $x_1 = x_2 = 1$. The $\vee$ operation is replaced by addition such that $x_1 + x_2 = 0$ if and only if $x_1 = x_2 = 0$. Evaluating the resulting polynomial over inputs $x_1, \ldots, x_k$ results in the message 0 if the boolean formula is 0, and a non-zero result otherwise. Utilizing this fact, a simple two party protocol for computing 2-DNF formulas on secret inputs, secure in the honest-but-curious model, is trivial to construct.

Protocol: Honest-But-Curious 2-DNF [14]
Setup: Alice possesses a 2-DNF formula $\phi(x_1, \ldots, x_k)$, Bob possesses a set of inputs $(a_1, \ldots, a_k)$, and both parties have agreed upon a security parameter ϵ.

1. Bob performs the following:
   (a) Create an instance of the BGN cryptosystem using security parameter ϵ, and sends the public key to Alice.
   (b) Compute $x_i = e(pk, a_i)$ for each input a_i, and sends each x_i to Alice.
2. Alice performs the following:
   (a) Compute the arithmetization Φ of ϕ by replacing $\wedge$ with $\cdot$, replacing $\vee$ with $+$, and replacing $\bar{x}_i$ with $(1 - x_i)$.
   (b) Using the homomorphic properties of the BGN cryptosystem, compute an encryption of $r\Phi(a_1, \ldots, a_k)$, sending the result to Bob.
3. If Bob receives an encryption of 0, he outputs "0"; otherwise, he outputs "1".

Alice is unable to learn anything about Bob's inputs without breaking the semantic security of the BGN cryptosystem. Bob is unable to learn anything about ϕ, aside from its evaluation on his inputs. This follows from the fact that Alice multiplies the result by a random value, which leaves Bob with either an encryption of 0, or an encryption of a random value.

The 2-DNF protocol can be modified to be secure against a malicious Bob. This is accomplished by requiring Bob to provide proof that he selected the parameters of the cryptosystem correctly, that he can decrypt messages encrypted under the public key he provides, as well forcing Bob's messages to be encryptions of only 0 or 1.

Protocol: Malicious 2-DNF [14]
Setup: Alice possesses a 2-DNF formula $\phi(x_1, \ldots, x_k)$, Bob possesses a set of inputs $(a_1, \ldots, a_k)$, and both parties have agreed upon a security parameter ϵ.

1. Bob creates an instance of the BGN cryptosystem using security parameter ϵ, and sends the public key to Alice. Bob also includes a zero-knowledge proof that n is the product of two primes.
2. Alice verifies that Bob has created a proper instance of the cryptosystem:
   (a) Alice verifies Bob's zero-knowledge proof, aborting if the proof is invalid.
   (b) Alice verifies that $g^n = h^n = 1$ and that $g, h \neq 1$, aborting if either test fails.
3. Bob proves that he can decrypt messages encrypted under the public key provided:
   (a) Alice selects random bits $b_0, \ldots, b_\epsilon$ and sends $e(pk, b_i)$ for each bit to Bob.
   (b) Bob replies with the decrypted values $b_0', \ldots, b_\epsilon'$.
   (c) Alice verifies that $b_i = b_i'$ for each i, aborting if any fail.
4. Bob computes $x_i = e(pk, a_i)$ for each input a_i, and sends each x_i to Alice.
5. Alice performs the following:
   (a) Compute the arithmetization Φ of ϕ.
   (b) Utilizing the homomorphic properties of the BGN cryptosystem, computes an encryption of

   $$r\Phi(x_1, \ldots, x_k) + \sum_{i=1}^{k} r_i x_i (x_i - 1)$$

   where r and each r_i are random values, and returns the result to Bob.
6. If Bob receives an encryption of 0, he outputs "0"; otherwise, he outputs "1".

As noted earlier, the three main changes from the honest-but-curious protocol are:

1. Forcing Bob to prove that the cryptosystem parameters have been chosen appropriately;
2. forcing Bob to prove he can decrypt messages encrypted under the provided public key; and

3. randomly blinding the result if Bob submits an input that is not an encryption of 0 or 1.

The proof of decryption in Step 3 is straightforward, and the resulting encrypted message in Step 5 will be a random value if $x_i(x_i - 1)$ is not 0 for all i; i.e., if $x_i \notin \{0, 1\}$. The zero-knowledge proof required in Step 2 can be accomplished through the protocol of Camenisch and Michels [19], or by that of Genarro, Micciancio, and Rabin [57]. With these restrictions placed on Bob, his behaviour is forced to be the same as in the honest-but-curious protocol, otherwise Alice aborts immediately.

5.1.2 Modeling Sets as Encrypted Polynomials

Polynomials lend themselves naturally to representing sets of integers. Given the set $S = \{s_1, s_2, \ldots, s_k\}$, the polynomial $f = (x - s_1)(x - s_2) \cdots (x - s_k)$ has zeros at $x = s_i$ for each $s_i \in S$. Given two sets S_1 and S_2 represented by polynomials f_1 and f_2, the polynomial $f_1 f_2$ has zeros at $x = a$ for all $a \in S_1 \cup S_2$. Similarly, if $a \in S_1 \cap S_2$, then $(x - a)^2 \,|\, f_1 f_2$ and $(x - a) \,|\, (f_1 f_2)'$; i.e., $a \in S_1 \cap S_2$ implies that a is a zero of $(f_1 f_2)'$.

Several protocols have been built on the idea of using encrypted polynomials to calculate simple set operations. Freedman, Nissim, and Pinkas [49] investigated the problem of two-party privacy preserving set intersection, referred to as private matching. In their protocol, a client $\mathcal{C}$ having set $C = \{c_1, \ldots, c_k\}$ wishes to determine which values it has in common with a server $\mathcal{S}$ having set $S = \{s_1, \ldots, s_j\}$.

Protocol: Honest-But-Curious Private Matching [49]
Setup: $\mathcal{C}$ and $\mathcal{S}$ possess sets C and S respectively, whose elements are drawn from a domain of size N.

1. $\mathcal{C}$ performs the following:
    (a) Create an instance of an additively homomorphic public-key cryptosystem, providing the homomorphic operations of $\boxplus$ and $\boxtimes_c$, whose plaintext space is exponentially larger than N. The public key, pk, is made public.
    (b) Calculate the polynomial $f = \prod_{i=1}^{k}(x - c_i) = f_0 + f_1 x + \ldots + f_k x^k$.
    (c) Send $\{E(pk, f_0), E(pk, f_1), \ldots, E(pk, f_k)\}$ to $\mathcal{S}$.
2. For each $s_i \in S$, $\mathcal{S}$ performs the following:
    (a) Using the homomorphic properties of the cryptosystem, compute the value $E(pk, f(s_i))$.
    (b) Choose a random value r and compute $m_i = E(pk, r \cdot f(s_i) + s_i)$.
    (c) Randomly permute the m_i's and sends the result to $\mathcal{C}$.

3. $\mathcal{C}$ decrypts each m_i. If the result is $s_i \in C$, then $s_i \in S \cap C$; otherwise m_i decrypts to a random value and $\mathcal{C}$ learns nothing.

The protocol works by randomly blinding any elements of the server's set whenever $f(s_i) \neq 0$. Thus, the client learns only those values which lie in the intersection. Because the client is assumed to be honest, the client does not attempt to learn additional information about the server's set by sending false input. A sample implementation of this protocol is provided in Appendix A.

Freedman, Nissim, and Pinkas also provide two variants of their set intersection, one secure against a malicious client, and one secure against a malicious server. Ideas for combining the two protocols together into a protocol secure against both a malicious client and server are also included. They also propose the problem of *fuzzy matching*, where set elements are tuples $(a_1, \ldots, a_m)$, and two parties are interested in determining which tuples agree on at least k-of-m entries. A naive approach to solving the problem is proposed, but the construction of a more efficient protocol is left as an open problem. Chmielewski and Hoepman [30] have since provided a more efficient approach.

Using an approach similar to Freedman, Nissim, and Pinkas, Frikken [50] has given a simple protocol for two party set union.

Protocol: Honest-But-Curious Set Union [50]
Setup: Participants P_1 and P_2 possess sets S_1 and S_2 respectively represented by the polynomials f_{S_1} and f_{S_2}.

1. P_1 creates an instance of an additively homomorphic cryptosystem.
2. P_1 encrypts each coefficient of the polynomial representation of S_1 and sends the resulting encrypted polynomial to P_2 along with the public key of the cryptosystem.
3. For each $s \in S_2$, P_2 chooses a random value r and computes the following tuple using the homomorphic properties of the cryptosystem
$$(E(pk, f_{S_1}(s)sr), E(pk, f_{S_1}(s)r)) \, .$$
4. P_2 randomly permutes the tuples and sends them to P_1.
5. P_1 sets $S = S_1$ and decrypts all received tuples. For each tuple (x, y), if $x = y = 0$, P_1 moves on to the next tuple; otherwise, P_1 adds the value xy^{-1} to S. Upon completion, $S = S_1 \cup S_2$.

This protocol is a very simple modification of the two party set intersection protocol. Instead of randomly blinding those values which are not in the intersection, a tuple is returned that allows P_1 to recover any set element not contained in the intersection. Thus, P_1 learns nothing about which elements are in the intersection, aside from the number of 0-tuples returned. P_2 could easily include extra "dummy tuples" encoding $(0, 0)$ to prevent P_1 from learning the cardinality of the intersection. A sample implementation of this protocol is provided in Appendix A.

Kissner and Song [67, 66] have used polynomials to represent multi-sets, or sets which may contain repeated elements. This is accomplished by using the factor $(x - a)^k$ to represent an element that occurs k times. Their protocols work for any number of participants, rather than just a single client and server, and provide protocols for computing multi-party set intersection, union, over-threshold union, intersection cardinality, and testing subsets. Their protocols are secure in the HBC model, with a variant of the set intersection protocol secure in the malicious model, although Frikken [50] has extended their set union protocol to be secure against malicious adversaries. Li and Wu [71] have also given a protocol for set intersection, and Ye, Wang, Pieprzyk, and Zhang [103] have presented a set disjointness test based on calculating the determinant of the Sylvester matrix.

5.2 A Limited Algebraically Homomorphic Cryptosystem

Sander, Young, and Yung's AND-homomorphic cryptosystem was presented in Section 3.2.6. Their approach utilized the Goldwasser-Micali cryptosystem, but added additional structure by representing messages as a vector of ciphertexts. The structure of the vector could be manipulated by a third party to compute the AND function on a ciphertext, with the semantic security GM cryptosystem keeping the message hidden. Sander, Young and Yung have used similar ideas to build a method of computing any NC^1 circuit on a set of encrypted inputs [90]. An NC^1 circuit is a log depth circuit that can be visualized as a binary tree. Each node of the tree functions as either an OR gate or a NOT gate, with inputs being provided to each gate at the bottom of the tree. The output of the circuit is the value output by the gate at the root of the tree.

Let x and y be binary values and consider the tuple $[x, y, x \oplus y, 1]$. If x OR $y = 1$, then this tuple contains three 1s, otherwise it contains a single 1. If a tuple with three 1s is considered an encoding of 1, and a tuple with three 0s is considered an encoding of 0, then the mapping $(x, y) \rightarrow [x, y, x \oplus y, 1]$ represents an encoding of $x + y$. If the tuple is randomly permuted after performing this operation, and contains an encoding of 1, then the tuple itself does not reveal the values of x and y. If $x = y = 0$, then the values are revealed by the result. If a tuple $[a, b, c, d]$ is of this form, then $[a \oplus 1, b \oplus 1, c \oplus 1, d \oplus 1]$ transforms a tuple with three 1s into three 0s, and a tuple with three 0s into a tuple with three 1s, effectively performing the NOT operation. Alternatively, if the bit is represented by the tuple $(x, \bar{x})$, then the NOT operation can be accomplished by replacing the tuple with $(\bar{x}, x)$, eliminating the need to perform four arithmetic operations.

Using the structure of these tuples, if each component is encrypted under a semantically secure cryptosystem implementing addition modulo 2 as a homomorphic operation, such as the Goldwasser-Micali cryptosystem, then Alice could send encrypted inputs to Bob, who could create tuples representing the NOT and OR operations on her inputs. Bob could also include inputs of his own. In order to build more complicated circuits the same idea is applied inductively, such that a tuple one level deeper in the circuit contains four tuples from the previous level. This process would be straightforward, if not for the fact that addition needs to be defined for tuples deeper in the circuit.

Let Enc_b^k represent an encoding of bit value b at level k of the circuit, and let $Enc^k = Enc_0^k \cup Enc_1^k$. Enc^0 represents inputs to the circuit provided by Alice and Bob. The set Add^k will represent the sum of two elements of Enc^{k-1}, stored as a tuple (a, b) to represent $a + b$. Then $(a, 0)$ where $a \in Enc^k$ maps a into Add^{k+1}. Given a starting point, each of these sets can be defined inductively:

$$\begin{aligned}
Add_0^{k+1} &= \{(a,b) \in (Enc^k)^2 \ : \text{both or none of } a, b \in Enc_0^k\} \\
Add_1^{k+1} &= \{(a,b) \in (Enc^k)^2 \ : \text{exactly one of } a, b \in Enc_0^k\} \\
Add^{k+1} &= Add_0^{k+1} \cup Add_1^{k+1} \\
Enc_0^{k+1} &= \{[a,b,c,d](Add^k)^4 \ : \text{exactly one of } a, b, c, d \in Enc_1^k\} \\
Enc_1^{k+1} &= \{[a,b,c,d](Add^k)^4 \ : \text{exactly three of } a, b, c, d \in Enc_1^k\} \\
Enc^{k+1} &= Enc_0^{k+1} \cup Enc_1^{k+1}.
\end{aligned}$$

Given these tuples, simple algorithms to encode and decode elements at each level can be defined, along with algorithms to perform NOT, OR, and to randomize a tuple to hide the values that created it. Given $x, y \in Enc^k$, $OR(x, y) = [(x, 0), (y, 0), (x, y), 1]$ where 1 is a random element of Add_1^{k+1}. Given $x = [(a_1, a_2), (b_1, b_2), (c_1, c_2), (d_1, d_2)]$, $NOT(x) = [(NOT(a_1), a_2), \ldots, (NOT(d_1), d_2)]$. Thus, the NOT operation recursively calls itself until it reaches the original encodings of Alice and Bob's inputs, allowing the homomorphic properties of the GM cryptosystem to be used to implement the final step. The decode method works in a similar manner. Given $(x, y) \in Add^{k+1}$, $DECODE((x, y)) = DECODE(x) + DECODE(Y)$. Randomizing a tuple is accomplished by computing a list of the possible tuples that decode to the same value, and randomly selecting one.

Given these sets and algorithms, Bob can take his and Alice's encrypted inputs and build an element of Enc^k that represents the evaluation of any NC^1 circuit on the inputs, without learning the value of the evaluation. Bob then sends the result to Alice, who can use the $DECODE$ method to recover the element of Enc^k without learning any more about the circuit than the result itself implies.

Although this approach allows for the joint computation of a wide variety of functions, its uses are limited by severe message expansion. Each bit of the input is encrypted under the Goldwasser-Micali cryptosystem, requiring a ciphertext expansion of 1024 times or larger. This is compounded by the fact that at each level of the circuit, a bit is represented by a four-tuple from the previous level.

Thus, there is an exponential growth in the size of representation as the circuit is evaluated. In most cases a protocol built specifically to solve a given problem will be much more efficient. Such a protocol is given in the next section for computing the greater than predicate, and subsequently improved to be more secure.

5.3 Strong Conditional Oblivious Transfer

Calculating the greater than predicate on two values arises naturally in many protocols, such as determining the winner of a private auction, or determining distances in location privacy protocols. Its use is often in the form of a *conditional oblivious transfer (COT)* protocol, where two parties, the sender and receiver, have inputs x and y, as well as a public predicate P, and the sender wishes to send a message m to the receiver if and only if $P(x, y) = 1$, while revealing no information if $P(x, y) = 0$. The sender does not learn the value of the predicate on the inputs; i.e., the sender is oblivious to which message the receiver receives. Under this definition, S always ends up learning the outcome of the predicate. In some applications it is desirable to for the sender to choose two messages m_1 and m_2, such that the receiver learns m_1 if $P(x, y) = 1$ and learns m_2 if $P(x, y) = 0$, and both parties remain oblivious to the actual evaluation of the predicate. A protocol implementing such functionality is said to be a *strong conditional oblivious transfer (SCOT)* protocol.

Blake and Kolesnikov [11] have constructed a SCOT protocol implementing the greater than predicate, constructed using the Paillier cryptosystem, and secure in the honest-but-curious model. Because their protocol relies only on the fact that the underlying cryptosystem is semantically secure and additively homomorphic, a different cryptosystem, such as the BGN cryptosystem, may be used in place of the Paillier cryptosystem. Utilizing this fact, a new protocol that uses the special properties of the BGN cryptosystem can be derived from the protocol of Blake and Kolesnikov, which adds security against a malicious receiver.

Let R be the receiver with input x, and S the sender with input y, where x and y are elements of some public input domain D_I whose elements have bit-length n. Let s_0 and s_1 be two messages chosen from a public domain D_S possessed by S. The sets D_I and D_S will be subsets of the plaintext space of the Paillier cryptosystem. Let ν and λ be public security parameters agreed upon by S and R such that $|D_S|$ is negligible with respect to λ, and let x_i (respectively y_i) denote the i'th most significant bit of x (respectively y).

Protocol: GT-SCOT [11]

1. R creates an instance of the Paillier cryptosystem using security parameter $\epsilon = \max\{\nu, \lambda + \log_2 |D_S|\}$ with public key (N, g) and creates the encrypted (component-wise) vector $(x_1, x_2, \ldots, x_n)$, sending it to S.
2. Using the homomorphic properties of the cryptosystem, for each $i \in \{1, \ldots, n\}$ S calculates:

(a) An encrypted element of the difference vector d such that $d_i = x_i - y_i$.

(b) An encrypted element of the flag vector f such that $f_i = x_i \oplus y_i = x_i - 2x_iy_i + y_i$.

(c) An encrypted element of the vector γ where $\gamma_0 = 0$ and $\gamma_i = 2\gamma_{i-1} + f_i$.

(d) An encrypted element of the vector δ where $\delta_i = d_i + r_i(\gamma_i - 1)$ where r_i is a random element of $\mathbb{Z}_N$.

(e) A random encryption of vector μ such that $\mu_i = \frac{s_1 - s_0}{2}\delta_i + \frac{s_1 - s_0}{2}$.

3. S randomly permutes μ and sends the result to R.

4. R decrypts each component of μ and checks if $\mu_i \in D_S$ for exactly one value of i. If so, R outputs μ_i, otherwise R aborts.

The protocol works by linearly searching from the most significant bit to the least significant bit for the first bit position that x and y differ. The flag vector has the value 1 at any position in which x and y differ, and d_i stores which of the two inputs contained the 1. The vector γ begins with initial value 0, and $\gamma_i = 0$ for each value of i until $f_i = 1$, at which point $\gamma_i = 1$. For $j > i$, it always holds that $\gamma_j > 1$. The vector δ contains a random value except when $\gamma_i = 1$; i.e., except in the position where x and y first differ, where the value d_i is stored. Finally, the vector μ contains a random value in each position, except where $\delta_i = d_i$. If $d_i = x_i - y_i = 1$, then $\mu_i = \frac{s_1 - s_0}{2} + \frac{s_1 + s_0}{2} = s_1$, and if $d_i = -1$, then $\mu_i = s_0$. S randomly permutes μ thus obscuring the position where x and y first differed, and sends the result to R. Because there is only a negligible probability for a random element of $\mathbb{Z}_N$ to be an element of D_S, with overwhelming probability exactly one elements of μ is also an element of D_S.

The GT-SCOT protocol only functions correctly if $x \neq y$ and R submits valid encryptions of either 0 or 1 in each position of the vector. If R is malicious, then R could attempt to construct each x_i in a manner that would reveal information about S's input when the protocol finishes. If the protocol is reimplemented using the BGN cryptosystem, then the additional multiplicative homomorphic property can be utilized by S to protect against a malicious R. These same ideas were used by Boneh, Goh, and Nissim, in the Malicious 2-DNF protocol described in Section 5.1.1.

Assuming the same setup as the GT-SCOT protocol, with the Paillier cryptosystem replaced by the BGN cryptosystem, the new Malicious Receiver GT-SCOT protocol can now be described.

Protocol: Malicious Receiver GT-SCOT

1. R creates an instance of the BGN cryptosystem using security parameter ϵ, and sends the public key $(N, g, h, \mathbb{G}, \mathbb{G}_1, e)$ to S. R also includes a zero-knowledge proof that N is the product of two primes.

2. S verifies that R has created a proper instance of the cryptosystem:
   (a) S verifies R's zero-knowledge proof, aborting if the proof is invalid.
   (b) S verifies that $g^n = h^n = 1$ and that $g, h \neq 1$, aborting if either test fails.
3. R proves that he can decrypt messages encrypted under the public key provided:
   (a) S selects random bits $b_0, \ldots, b_\epsilon$ and sends $e(pk, b_i)$ for each bit to R.
   (b) R replies with the decrypted values ${b_0}', \ldots, {b_\epsilon}'$.
   (c) S verifies that $b_i = {b_i}'$ for each i, aborting if any fail.
4. R sends the encrypted (component-wise) vector $x' = ({x_1}', {x_2}', \ldots, {x_n}')$ to S.
5. S calculates, using the multiplicative homomorphic properties of the BGN cryptosystem, the encrypted vector x such that $x_i = {x_i}' + r_i {x_i}'({x_i}' - 1)$ where r_i is a random value in $\mathbb{Z}_N$.
6. S continues from step 2 of the GT-SCOT protocol taking x to be R's input.

In this new variant of the GT-SCOT protocol, R and S perform the same verification steps as Alice and Bob performed in the Malicious 2-DNF protocol. Once S is convinced that R has created the cryptosystem correctly, S receives a component-wise encrypted vector from R. In order to ensure that R has selected $x_i \in \{0, 1\}$, S adds the value $r_i {x_i}'({x_i}' - 1)$, randomizing R's input if it is not a valid bit. Thus, R cannot submit a malicious message without S transforming it into a completely random message.

It should be noted that the GT-SCOT protocol does not function correctly if $x = y$. In order to prevent this from happening, R's input x can be mapped to $2x$, and S's input y can be mapped to $2y + 1$, thus ensuring that $x \neq y$. In a setting where R is untrusted, S can perform the mapping by "bit shifting" R's input by a single bit and appending a random encryption of 0 to the end.

5.4 Cryptographic Voting and Mix Nets

The Benaloh cryptosystem was originally presented by Cohen (Benaloh) and Fischer [35] in the context of a cryptographic election scheme where a government G and a set of voters $v_1, \ldots, v_j$ wish to conduct an election where the ballot contains a single yes/no question. The Benaloh ciphertext $c = y^0 u^r$ represents an encrypted vote for "no", and the ciphertext $c = y^1 u^r$ represents an encrypted vote for "yes". A blank ballot from voter i consists of a pair $B_i = (c_0, c_1)$ where one of c_0 and c_1 is a "yes" vote, and the other is a "no" vote. To conduct the election, G posts a set of public parameters along with proofs that the parameters are correct. Each voter constructs a blank ballot using these public parameters and proves in zero-knowledge that it contains both a “yes” vote and a “no” vote, without revealing which vote is which. Each voter then chooses a single value from the blank ballot as their choice and sends it to G, who verifies the proof and uses the homomorphic properties of the cryptosystem to calculate a tally of the yes votes, along with a proof that the tally is correct. Details of the necessary proofs can be found in [35].

Adida and Rivest use the homomorphic properties of the Paillier cryptosystem to perform a homomorphic tally in the Scratch & Vote system [1]. Given $M < 2^m$ voters and j candidates numbered $0, \ldots, j-1$, an instance of the Paillier cryptosystem is created such that $n > 2^{jm}$. Thus, the plaintext space can be thought of as j different m-bit numbers concatenated together, with the i'th least significant block of m-bits representing the tally for candidate i. On each ballot, candidate i is listed along with a random ciphertext encrypting the value $2^i m$, corresponding to a value of 1 in that candidates tally. The candidate ordering on each ballot is randomized, and the list of candidates is detachable. In order to cast a vote, the voter simply marks the ciphertext beside their choice, and tears off the list of candidates, thus removing any public information on which candidate was marked. The homomorphic properties of the Paillier cryptosystem are used to sum all ballots, and the final result is jointly decrypted by election officials.

When conducting elections with electronic votes, a digital analog to “shaking the ballot box” is necessary to remove any link between a ballot and the voter who submitted it. Mix nets, introduced by Chaum [26], provide this functionality. Adida and Wikström [2, 3] have demonstrated two different approaches of shuffling encrypted ballots based on the Paillier and BGN cryptosystems. Let π be a random permutation, and let M be a matrix that applies π to a vector v. Given a public vector of encrypted ballots, the goal is to allow any party to apply and verify the permutation, resulting in a new vector of re-randomized ciphertexts, without revealing the permutation.

Recall that the BGN allows a single homomorphic multiplication on ciphertexts, and thus, a single encrypted matrix multiplication is possible. Election officials may produce a permutation matrix and encrypt each element of the matrix as a randomized BGN encryption of either 0 or 1, along with zero knowledge proofs that the matrix is constructed correctly. By the semantic security of the BGN cryptosystem, the matrix does not reveal the permutation. Anybody may utilize the homomorphic properties of the BGN cryptosystem to apply the permutation to the encrypted vector of ballots, which are re-randomized automatically, but deterministically, as the permutation is applied. This is due to the fact that the randomizers used to encrypt the matrix entries are not public.

The Paillier cryptosystem lacks the single homomorphic multiplication of the BGN cryptosystem, but has another interesting property that can be utilized to perform a public shuffle. Recall that the Paillier cryptosystem maps a plaintext in $\mathbb{Z}_n$ to a ciphertext in $\mathbb{Z}^*_{n^2}$ and the Damgård-Jurik variant maps a plaintext in $\mathbb{Z}_{n^s}$ to a ciphertext in $\mathbb{Z}^*_{n^{s+1}}$. Thus, Paillier ciphertexts are valid plaintexts in the Damgård-Jurik cryptosystem, and both cryptosystems support the same homomorphic operations. If E performs a random encryption in an instance of the Paillier cryptosystem, and E' performs a random encryption in the Damgård-Jurik cryptosystem for the same choice of n and $s = 2$, then the following hold:

$$\begin{aligned} E'(1)^{E(m)} &= E'(E(m)) \\ E'(0)^{E(m)} &= E'(0) \\ E'(0)E'(E(m)) &= E'(E(m)). \end{aligned}$$

Election officials can construct a permutation matrix, and then replace each 0 in the matrix with $E'(0)$, and each 1 with $E'(E(0))$ before making it public. The encrypted ballots are encrypted using E. The equations above can then be used to apply the permutation matrix to the encrypted vector, resulting in a vector $\pi(v)$ containing deterministically randomized ballots of the form $E'(E(v_{\pi(i)}))$ for each $v_{\pi(i)} \in v$.

Additional details on zero-knowledge proofs that the election officials have constructed each ciphertext and the matrix appropriately may be found in [2, 3].

Chapter 6

Concluding Remarks

Semantically secure homomorphic cryptography began with the Goldwasser-Micali cryptosystem, allowing the XOR operation on ciphertexts, with each bit of a message encrypted as its own ciphertext. From this starting point, a variety of additively homomorphic cryptosystems have been based on the approach of Goldwasser and Micali, exploiting the difficulty of deciding residues in various settings. Modern additively homomorphic cryptosystems have decreased the unrealistic message expansion of the Goldwasser-Micali cryptosystem to a constant factor of 2, while still remaining relatively efficient.

In this thesis, a comprehensive survey of homomorphic cryptosystems has been presented, describing each additively homomorphic cryptosystem building on Goldwasser and Micali's approach, as well as other homomorphic cryptosystems, such as the multiplicatively homomorphic ElGamal cryptosystem. In each case, the underlying computational problems have been presented, alongside outlines of security proofs.

Because homomorphic cryptography is ideal for use in privacy preserving protocols, it is often desirable to devise threshold variants of homomorphic cryptosystems. This allows for the private key to be distributed among a group of participants such that at least t of n participants must agree to decrypt a message. A survey of threshold homomorphic cryptosystems has been provided, and the approaches used to build these threshold variants have been applied to the recently discovered Boneh-Goh-Nissim cryptosystem to define the first secure threshold variant. This threshold variant builds off the well studied approach of Shoup for distributed RSA signature generation, and is shown to be secure under the threshold semantic security game of Fouque, Poupard, and Stern.

Due to the fact that it allows a single homomorphic multiplication, in addition to the usual properties of an additively homomorphic cryptosystem, the BGN cryptosystem is of particular interest. In particular, the multiplicative homomorphism may be used to check whether or not an encrypted input is one of two known values, randomizing it if this is not the case. This provides a very simple mechanism for creating protocols that automatically randomize maliciously crafted messages. This fact was exploited to create a variant of Blake and Kolesnikov's greater than strong conditional oblivious transfer protocol that is secure against a malicious receiver.

The wide variety of applications of homomorphic cryptography make it difficult to provide a comprehensive survey, so the examples presented were limited to provide a basic idea of how homomorphic cryptography can be used as a building block in privacy preserving protocols. In order to demonstrate the correctness of these protocols, a sample implementation of Freedman, Nissim, and Pinkas' private matching protocol, and Frikken's HBC set union protocol have been provided in Appendix A.

6.1 Future Work

Although homomorphic cryptosystems have advanced considerably since the Goldwasser-Micali cryptosystem, most of the advancements have been with respect to message expansion and efficiency. Only recently, with the discovery of the Boneh-Goh-Nissim cryptosystem, has any significant advance towards a secure algebraically homomorphic cryptosystem been made. Although the approach used by Boneh, Goh, and Nissim is unlikely to lead to a cryptosystem allowing an unlimited number of homomorphic multiplications, the ability to construct n-linear maps for cryptographic purposes would allow their approach to be used to perform n homomorphic multiplications. This would allow for the evaluation of n-dnf formulas on ciphertexts, as well as the solution to other problems that can be modeled as multivariate polynomials with terms of degree n or less. This would also allow for a user to randomize an encrypted message if it is not an element of a specified set of n messages, an application that could be useful in cryptographic voting protocols.

The creation of a secure algebraically homomorphic cryptosystem is the most prominent open problem in homomorphic cryptography. Although some theoretical attacks against such a cryptosystem have been proposed, the question of whether or not a secure algebraically homomorphic cryptosystem exists has yet to be answered. One promising approach is the investigation of homomorphic cryptosystems defined over non-abelian groups, as no such cryptosystem is currently known to exist. This question was asked directly by Sander, Young, and Yung [90].

The creation of threshold variants of the Goldwasser-Micali, Naccache-Stern, and Okamoto-Uchiyama cryptosystems is still an open problem, and a solution may have intresting applications in other protocols. In the case of the Naccache-Stern cryptosystem, a protocol that distributes Chinese remaindering modulo a set of secret values is required, while the GM and OU cryptosystem simply require a single reduction modulo a secret value.

Although homomorphic cryptosystems have already found several applications in privacy preserving protocols, there are likely many more applications to be found, and many older protocols can be made more efficient or secure by utilizing more recent cryptosystems with additional homomorphic properties, such as the BGN cryptosystem. In general, single purpose protocols designed using homomorphic cryptography are more efficient than general approaches. As homomorphic cryptosystems with new properties are discovered, many protocols will need to be re-evaluated to determine whether or not they can benefit from these new properties.

References

[1] Ben Adida and Ronald L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In Ari Juels and Marianne Winslett, editors, *WPES 2006*, pages 29–40. ACM, 2006. 107

[2] Ben Adida and Douglas Wikström. How to shuffle in public. Cryptology ePrint Archive, Report 2005/394, 2005. `http://eprint.iacr.org/`. 107, 108

[3] Ben Adida and Douglas Wikström. How to shuffle in public. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 555–574. Springer, 2007. 107, 108

[4] Niv Ahituv, Yeheskel Lapid, and Seev Neumann. Processing encrypted data. *Commun. ACM*, 30(9):777–780, 1987. 15

[5] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002. 18

[6] Feng Bao. Cryptanalysis of a provable secure additive and multiplicative privacy homomorphism. In *International Workshop on Coding and Cryptography, March 24-28, 2003, Versailles (France)*, pages 43–50, 2003. 68

[7] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994. 6

[8] Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, 1988. 39

[9] Josh Benaloh. Dense probabilistic encryption. In *Selected Areas of Cryptography (SAC 1994)*, pages 120–128, 1994. 38, 39

[10] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, July 1999. 31

[11] Ian F. Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In Pil Joong Lee, editor, *ASIACRYPT*, volume

3329 of *Lecture Notes in Computer Science*, pages 515–529. Springer, 2004. 104

[12] G.R. Blakley. Safeguarding cryptographic keys. In *AFIPS 1979 Conference Proceedings*, volume 48, pages 313–317, 1979. 71

[13] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. 53

[14] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005. 2, 31, 66, 67, 97, 98, 99

[15] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Koblitz [68], pages 283–297. 16, 17

[16] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Nyberg [75], pages 59–71. 32

[17] Ernest F. Brickell and Yacov Yacobi. On privacy homomorphisms (extended abstract). In David Chaum and Wyn L. Price, editors, *EUROCRYPT 1987*, volume 304 of *Lecture Notes in Computer Science*, pages 117–125. Springer, 1987. 11

[18] Dan Brown. Breaking RSA may be as difficult as factoring. Technical Report CACR 2005-37, Center for Applied Cryptographic Research (CACR): University of Waterloo, 2005. 33

[19] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In Stern [97], pages 107–122. 100

[20] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003. 19

[21] Guilhem Castagnos. An efficient probabilistic public-key cryptosystem over quadratic fields quotients. *Finite Fields Appl.*, 13(3):563–576, 2007. 56

[22] Guilhem Castagnos and Benoît Chevallier-Mames. Towards a dl-based additively homomorphic encryption scheme. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC 2007*, volume 4779 of *Lecture Notes in Computer Science*, pages 362–375. Springer, 2007. 37

[23] Dario Catalano, Rosario Gennaro, and Nick Howgrave-Graham. The bit security of paillier's encryption scheme and its applications. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2001. 53

[24] Dario Catalano, Rosario Gennaro, and Nick Howgrave-Graham. Paillier's trapdoor function hides up to $O(n)$ bits. *J. Cryptology*, 15(4):251–269, 2002. 53

[25] Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q. Nguyen. Paillier's cryptosystem revisited. In *ACM Conference on Computer and Communications Security*, pages 206–214, 2001. 25, 56

[26] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981. 107

[27] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992. 76, 83, 84

[28] Jung Hee Cheon, Woo-Hwan Kim, and Hyun Soo Nam. Known-plaintext cryptanalysis of the Domingo-Ferrer algebraic privacy homomorphism scheme. *Inf. Process. Lett.*, 97(3):118–123, 2006. 68

[29] Benoît Chevallier-Mames, Pascal Paillier, and David Pointcheval. Encoding-free ElGamal encryption without random oracles. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography (PKC 2006)*, volume 3958 of *Lecture Notes in Computer Science*, pages 91–104. Springer, 2006. 37

[30] Lukasz Chmielewski and Jaap-Henk Hoepman. Fuzzy private matching (extended abstract). In *ARES 2008*, pages 327–334. IEEE Computer Society, 2008. 101

[31] Dug-Hwan Choi, Seungbok Choi, and Dongho Won. Improvement of probabilistic public key cryptosystems using discrete logarithm. In Kwangjo Kim, editor, *ICISC 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 72–80. Springer, 2001. 50, 57

[32] Su-Jeong Choi. *Cryptanalysis of a Homomorphic Public-Key Cryptosystem.* PhD thesis, Royal Holloway University of London, 2006. 68

[33] Su-Jeong Choi, Simon R. Blackburn, and Peter R. Wild. Cryptanalysis of a homomorphic public-key cryptosystem over a finite group. *Journal of Mathematical Cryptography*, 1:351–358, 2007. 68

[34] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In Tarjan [99], pages 383–395. 73

[35] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In Tarjan [99], pages 372–382. 27, 39, 106, 107

[36] Jean-Sébastien Coron, David Naccache, and Pascal Paillier. Accelerating Okamoto-Uchiyama's public-key cryptosystem. *Electronics Letters*, 35(4):291–292, 1999. 50

[37] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Fumy [52], pages 103–118. 37, 83, 92

[38] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998. 37

[39] Ivan Damgård, Mads Jurik, , and Jesper Nielsen. A generalization of Paillier's public-key system with applications to electronic voting, 2003. 80, 81

[40] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography (PKC 2001)*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001. 60, 61, 80, 81

[41] Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. Technical Report RS/03/16, BRICS - University of Aarhus, 2003. 63, 64

[42] Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2003. 63, 64, 82

[43] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976. 4

[44] Josep Domingo-Ferrer. A new privacy homomorphism and applications. *Inf. Process. Lett.*, 60(5):277–282, 1996. 68

[45] Josep Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In Agnes Hui Chan and Virgil D. Gligor, editors, *ISC 2002*, volume 2433 of *Lecture Notes in Computer Science*, pages 471–483. Springer, 2002. 68

[46] Josep Domingo-Ferrer and Jordi Herrera-Joancomart. A privacy homomorphism allowing field operations on encrypted data. 16

[47] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *Eur. J. Inf. Syst.*, 2007(1):1–15, 2007. 23

[48] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In Yair Frankel, editor, *Financial Cryptography (FC 2000)*, volume 1962 of *Lecture Notes in Computer Science*, pages 90–104. Springer, 2000. 2, 77, 78, 80, 84, 91

[49] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004. 100

[50] Keith B. Frikken. Privacy-preserving set union. In Katz and Yung [65], pages 237–252. 101, 102

[51] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography (PKC 1999)*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999. 37

[52] Walter Fumy, editor. *Advances in Cryptology - EUROCRYPT 1997, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*. Springer, 1997. 115, 119

[53] Steven D. Galbraith. Elliptic curve Paillier schemes. *J. Cryptology*, 15(2):129–138, 2002. 32, 51, 57, 59

[54] Joseph A. Gallian. *Contemporary Abstract Algebra*. Houghton Mifflin, Boston, fifth edition, 2002. 29, 46

[55] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984. 35

[56] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust and efficient sharing of RSA functions. In Koblitz [68], pages 157–172. 93

[57] Rosario Gennaro, Daniele Micciancio, and Tal Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *ACM Conference on Computer and Communications Security*, pages 67–72, 1998. 100

[58] Rosario Gennaro, Tal Rabin, Stanislaw Jarecki, and Hugo Krawczyk. Robust and efficient sharing of RSA functions. *J. Cryptology*, 13(2):273–300, 2000. 93

[59] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC 1989: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, New York, NY, USA, 1989. ACM. 53

[60] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000. 43

[61] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC 1982*, pages 365–377. ACM, 1982. 6

[62] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. 6, 33

[63] Dima Grigoriev and Ilia V. Ponomarenko. Homomorphic public-key cryptosystems over groups and rings. *CoRR*, cs.CR/0309010, 2003. 68

[64] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. 34

[65] Jonathan Katz and Moti Yung, editors. *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, volume 4521 of *Lecture Notes in Computer Science*. Springer, 2007. 116, 117

[66] Lea Kissner and Dawn Song. Private and threshold set-intersection. Technical Report CMU-CS-05-113, Carnegie Mellon University, February 2005. 102

[67] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2005. 102

[68] Neal Koblitz, editor. *Advances in Cryptology - CRYPTO 1996, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*. Springer, 1996. 113, 116

[69] Arjen K. Lenstra and Hendrik W. Lenstra, editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993. 24

[70] Hendrik W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987. 18, 24

[71] Ronghua Li and Chuankun Wu. An unconditionally secure protocol for multi-party set intersection. In Katz and Yung [65], pages 226–236. 102

[72] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001. 2, 24

[73] Victor S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004. 32, 66

[74] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM Conference on Computer and Communications Security*, pages 59–66, 1998. 40, 43

[75] Kaisa Nyberg, editor. *Advances in Cryptology - EUROCRYPT 1998, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*. Springer, 1998. 113, 118

[76] Andrew M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *EUROCRYPT 1984*, volume 209 of *Lecture Notes in Computer Science*, pages 224–314. Springer, 1984. 30

[77] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In Nyberg [75], pages 308–318. 46, 47, 48, 49

[78] Tatsuaki Okamoto and Shigenori Uchiyama. Security of an identity-based cryptosystem and the related reductions. In Nyberg [75], pages 546–560. 51, 59

[79] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Stern [97], pages 223–238. 27, 29, 51, 53, 55

[80] Pascal Paillier. Trapdooring discrete logarithms on elliptic curves over rings. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 573–584. Springer, 2000. 51, 56, 59

[81] Pascal Paillier and David Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT 1999*, volume 1716 of *Lecture Notes in Computer Science*, pages 165–179. Springer, 1999. 56

[82] Renè Peralta and Eiji Okamoto. Faster factoring of integers of a special form. *IEICE Transactions on Communications/Electronics/Information and Systems*, E79-A(4):489–493, 1996. 24

[83] Manoj Prabhakaran and Mike Rosulek. Homomorphic encryption with CCA security. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 667–678. Springer, 2008. 20

[84] Manoj Prabhakaran and Mike Rosulek. Homomorphic encryption with chosen-ciphertext security. Cryptology ePrint Archive, Report 2008/079, 2008. `http://eprint.iacr.org/`. 20, 21

[85] Michael Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979. 49

[86] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzous. On data banks and privacy homomorphisms. In Richard. A. Demillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computations*, pages 169–177. Academic Press, New York, 1978. 1, 10

[87] Ronald L. Rivest and Burton S. Kaliski Jr. RSA problem. In Henk C. A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*. Springer, 2005. 25

[88] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. 4

[89] Kouichi Sakurai and Tsuyoshi Takagi. On the security of a modified paillier public-key primitive. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP 2002*, volume 2384 of *Lecture Notes in Computer Science*, pages 436–448. Springer, 2002. 50, 57, 58, 59

[90] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *FOCS 1999*, pages 554–567. IEEE, 1999. 44, 102, 110

[91] Katja Schmidt-Samoa and Tsuyoshi Takagi. Paillier's cryptosystem modulo p^2q and its applications to trapdoor commitment schemes. In Ed Dawson and Serge Vaudenay, editors, *Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 296–313. Springer, 2005. 57

[92] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. 71

[93] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. 18

[94] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Fumy [52], pages 256–266. 30

[95] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2000. 2, 73, 88

[96] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/`. 18, 76

[97] Jacques Stern, editor. *Advances in Cryptology - EUROCRYPT 1999, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*. Springer, 1999. 113, 118

[98] Douglas R. Stinson. *Cryptography: Theory and Practice.* Chapman & Hall/CRC, 2005. 3, 6

[99] Robert E. Tarjan, editor. *26th Annual Symposium on Foundations of Computer Science, 21-23 October 1985, Portland, Oregon, USA*. IEEE, 1985. 114

[100] E. Teske. Square-root algorithms for the discrete logarithm problem. In *Public Key Cryptography and Computational Number Theory*, pages 283–301. Walter de Gruyter, 2001. 30, 39

[101] Wim van Dam, Sean Hallgren, and Lawrence Ip. Quantum algorithms for some hidden shift problems. *SIAM J. Comput.*, 36(3):763–778, 2006. 18

[102] David Wagner. Cryptanalysis of an algebraic privacy homomorphism. In Colin Boyd and Wenbo Mao, editors, *ISC 2003*, volume 2851 of *Lecture Notes in Computer Science*, pages 234–239. Springer, 2003. 68

[103] Qingsong Ye, Huaxiong Wang, Josef Pieprzyk, and Xian-Mo Zhang. Efficient disjointness tests for private datasets. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 2008*, volume 5107 of *Lecture Notes in Computer Science*, pages 155–169. Springer, 2008. 102

[104] Yu Yu, Jussipekka Leiwo, and Benjamin Premkumar. A study on the security of privacy homomorphism. *International Journal of Network Security*, 6(1):33–39, 2008. 13, 14, 15, 16

[105] Ge Zhong. Distributed approaches for location privacy. Master's thesis, University of Waterloo, 2008. 94

[106] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, Lester and Pierre: Three protocols for location privacy. In Nikita Borisov and Philippe Golle, editors, *Privacy Enhancing Technologies (PET 2007)*, volume 4776 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2007. 94

Appendix A

Sample Implementations

In this section some sample implementations of the privacy preserving set union and private matching protocols from Section 5.1.2 are given, using Damgård and Jurik's generalization of the Paillier cryptosystem. The implementation is done in Maple 10, and the algorithms are implemented in a straightforward and easy to follow manner. For ease of reading, a very small security parameter is used, although the implementation provided can accommodate any security parameter.

A.1 A Sample Implementation of the Paillier / Damgård-Jurik Cryptosystem

Recall the Damgård-Jurik Cryptosystem from Section 3.2.11, which extends the Paillier cryptosystem to encrypt messages from the group $\mathbb{Z}_{n^s}$ to ciphertexts in $\mathbb{Z}^*_{n^{s+1}}$ for any $s \geq 1$. The log function $L(u) = \frac{u-1}{n}$ is replaced by an algorithm that recovers the value i from $L((1+n)^i \mod n^{s+1})$. The following code defines the function L and the procedure **cosetLog** that recovers i.

```
> L := (u,n,s) -> (u-1)/n:
>
> cosetLog := proc(u, sk)
>    local i, j, k, t1, t2, n, s:
>
>    n := pk[1]:
>    s := pk[3]:
>
>    i := 0;
>    for j from 1 to s by 1 do
>       t1 := L(u mod n^(j+1), n, s):
>       t2 := i;
>       for k from 2 to j by 1 do
>           i := i-1:
>           t2 := t2 * i mod n^j:
>           t1 := t1 - (t2 * n^(k-1))/(k!) mod n^j;
>        od:
>        i := t1;
>     od:
>
>     return i;
>
> end:
```

The public key **pk** is stored as the tuple $pk = (n, g, s)$, and the private key **sk** is stored as the tuple $sk = (\lambda, pk, p, q)$, where n, p, q, g, s, λ are as defined in the Damgård-Jurik cryptosystem. Given a security parameter and the value s, the procedure **Gen** creates an instance of the Damgård-Jurik cryptosystem and returns the public and private keys.

```
> gen := proc(epsilon, s)
>     local n, p, q, lambda, g, pk, sk, x, j;
>
>     p := nextprime( 2^epsilon +
  RandomTools[Generate](integer(range=0..2^epsilon))):
>     q := nextprime( 2^epsilon +
  RandomTools[Generate](integer(range=0..2^epsilon))):
>     n := p*q:
>     lambda := lcm(p-1, q-1):
>
>     x := rand() mod n:
>     j := rand() mod n:
>     g := (1+n)&^j * x mod n^(s+1):
>
>     pk := [n, g, s]:
>     sk := [lambda, pk, p, q]:
>
>     return [pk, sk]:
>
> end:
```

The procedure **Enc** takes a public key pk, and a message m, and returns a randomized encryption of m.

```
> enc := proc(pk, m)
>     local r, c, g, n, s:
>
```

```
>    n := pk[1]:
>    g := pk[2]:
>    s := pk[3]:
>
>    r := modp(rand(), n^s):
>
>    c := modp(g&^m * r&^(n^s), n^(s+1));
>
>    return c:
>
> end:
```

The procedure **Dec** takes a private key sk, and a ciphertext c and returns the decrypted message.

```
> dec := proc(sk, c)
>
>    local m, lambda, n, s, g, i, j, k, t1, t2:
>
>    lambda := sk[1]:
>    n := (sk[2])[1]:
>    s := (sk[2])[3]:
>    g := (sk[2])[2]:
>
>    m := cosetLog(c&^lambda mod n^(s+1))
     / cosetLog(g&^lambda mod n^(s+1)) mod n^s:
>
>    return m:
>
> end:
```

With the procedures **Gen**, **Enc**, and **Dec** defined, an instance of the cryptosystem can be created, and the proper decryption of messages can be tested. First, an instance of the cryptosystem using $\epsilon = 32$ and $s = 2$ is created:

```
> k := gen(32, 2):
> pk := k[1]:
> sk := k[2]:
> printf("\n\tn = %d \n\tp = %d \n\tq = %d \n
   \tlambda = %d \n\tg = %d \n\ts = %d\n\n",
   pk[1],sk[3],sk[4],sk[1],pk[2], pk[3]);

  n = 38435821667422746529
  p = 4876836619
```

```
q = 7881301891
lambda = 6405970275777434670
g = 1482205154450878409516275481912474757577699\
    144879486424051
s = 2
```

To demonstrate that encryption and decryption work, the same message is encrypted three times, resulting in three different ciphertexts. All three ciphertexts decrypt to the same original message.

```
> m := RandomTools[Generate](integer(range=0..pk[1]^(pk[3])));
> c1 := enc(pk, m);
> c2 := enc(pk, m);
> c3 := enc(pk, m);
> m1 := dec(sk, c1);
> m2 := dec(sk, c2);
> m3 := dec(sk, c3);

 m := 785428547153071673492364480495024318660
 c1 := 3633368313374378653990508384906491247014\
       4431932610902378615
 c2 := 4241022925979226859212233388882057998592\
       5359469096550718325
 c3 := 1008383705742527764693669736714639570150\
       7905959427821292317
 m1 := 785428547153071673492364480495024318660
 m2 := 785428547153071673492364480495024318660
 m3 := 785428547153071673492364480495024318660
```

To demonstrate the homomorphic properties of the cryptosystem, the messages $m1 = 100$ and $m2 = 25$ are encrypted as $c1$ and $c2$. The product $c1 * c2$ correctly decrypts to the sum $m1 + m2$, and $(c1 * c2)^5$ correctly decrypts to $5(m1 + m2)$.

```
> m1 := 100;
> m2 := 25;
> c1 := enc(pk, m1);
> c2 := enc(pk, m2);

 m1 := 100
 m2 := 25
 c1 := 5708135225555232216299419403706442583834\
       059880755561689895
 c2 := 2442211797371755904111254871307554957029\
       4512906778555547172
```

```
> dec(sk, c1 * c2);
 125

> dec(sk, (c1 * c2)^5 );
 625
```

Thus, the given implementation of the Damgård-Jurik cryptosystem appears correct, and can now be used to implement the private matching and set union protocols.

A.2 Private Matching

The two party HBC private matching protocol by Freedman, Nissim, and Pinkas was presented in Section 5.1.2. In this section, a sample run of the protocol is provided to demonstrate that it works in practice.

Assume the client C possesses the set $S_1 = \{1, 2, 3, 4, 5, 6\}$ and the Server S possesses the set $S2 = \{4, 5, 6, 7, 8, 9\}$. The domain of possible set elements is assumed to be $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Before running the protocol, two helper methods are required. The procedure **encPoly** takes a public key *pk* and a polynomial h and returns a vector with encryptions of each coefficient of h. The procedure **encEval** takes an encrypted polynomial, a public key, and a set element s, and returns the encrypted evaluation of the polynomial at s.

```
> encPoly := proc(pk, h)
>    local H, i:
>
>    H := Array(0..degree(h, x)):
>
>    for i from 0 to degree(h,x) by 1 do
>       H[i] := enc(pk, coeff(h,x,i) mod pk[1]^(pk[3]));
>    od:
>
>    return H:
>
> end:

> encEval := proc(pk, F, s)
>    local c, i:
>
>    c := 1:
>
>    for i from 0 to ArrayNumElems(F) - 1 by 1 do
>       c := c * F[i]&^(s^i):
>    od:
>
>    return c:
>
> end:
```

A.2.1 Step 1

During step 1 of the protocol, C performs the following:

Creates an instance of the cryptosystem:

```
> k := gen(16, 2):
> pk := k[1]:
> sk := k[2]:
```

Calculates the polynomial representation of S_1:

```
> S1 := [1,2,3,4,5,6]:
> f := 1:
> for i from 1 to 6 by 1 do
>    f := f * (x - S1[i]);
> od:
> f := sort(collect(f,x));
> f := f mod pk[1]^(pk[3]):
```

resulting in the polynomial

$$f = x^6 - 21x^5 + 175x^4 - 735x^3 + 1624x^2 - 1764x + 720.$$

C then encrypts each coefficient of f and sends the result to S:

```
> F := encPoly(pk, f):
> for i from 0 to ArrayNumElems(F) - 1 by 1 do
>    printf("\t F[%d] = %d\n", i, F[i]);
> od:

  F[0] = 7978474253566748455725491019
  F[1] = 205483925844442872149872396344
  F[2] = 20758809590907471195066201753
  F[3] = 336967061963213296775929596270
  F[4] = 144788316426894907426468516314
  F[5] = 358537103547234329395169107496
  F[6] = 48658219063554320500156480832
```

A.2.2 Step 2

During step 2 of the protocol, S performs the following for each $s_i \in S_2$:
Calculates an encryption of $f(s_i)$

```
> S2 := [4,5,6,7,8,9]:
> m := [0,0,0,0,0,0]:
> for i from 1 to 6 by 1 do
>    m[i] := encEval(pk, F, S2[i]) mod pk[1]^(pk[3] + 1):
>    printf("\tEnc(pk, f(%d)) = %d\n", S2[i], m[i]);
> od:

  Enc(pk, f(4)) = 390293989761110743213423828375
  Enc(pk, f(5)) = 177112131816332494500982198292
  Enc(pk, f(6)) = 302868852682525188991041682915
  Enc(pk, f(7)) = 436123105498581796228432884843
  Enc(pk, f(8)) = 455225100074763047240491268647
  Enc(pk, f(9)) = 116180799576401424702455543044
```

S then chooses a random value r and computes an encryption of $(r \cdot f(s_i) + s_i)$

```
> for i from 1 to 6 by 1 do
>    r := RandomTools[Generate](integer(range=0..pk[1])):
>    m[i] := m[i]&^r * enc(pk, S2[i]) mod pk[1]^(pk[3] + 1):
>    printf("\tEnc(pk, r * f(%d) + %d) = %d\n",
           S2[i], S2[i], m[i]);
> od:

  Enc(pk, r * f(4) + 4) = 478355773033542494059416337920
  Enc(pk, r * f(5) + 5) = 54423609613207795791742498284
  Enc(pk, r * f(6) + 6) = 284343525423413532338299278695
  Enc(pk, r * f(7) + 7) = 145197479142440469519970087896
  Enc(pk, r * f(8) + 8) = 46817530870050439866409604957
  Enc(pk, r * f(9) + 9) = 385373772527198658778423947649
```

The messages are randomly permuted and sent back to C:

```
> perm := combinat[permute](m):
> m := perm[rand() mod 6!]:
> for i from 1 to 6 by 1 do
>    printf("\tm[%d] = %d\n", i, m[i]);
> od:
```

```
m[1] = 54423609613207795791742498284
m[2] = 478355773033542494059416337920
m[3] = 284343525423413532338299278695
m[4] = 385373772527198658778423947649
m[5] = 46817530870050439866409604957
m[6] = 145197479142440469519970087896
```

A.2.3 Step 3

Upon receiving the messages from S, C decrypts each of them:

```
> for i from 1 to 6 by 1 do
>    print(dec(sk, m[i]));
> od:

                         5
                         4
                         6
                   157752301150089
                    18885148380008
                    5557866089047
```

and, because $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, C concludes that $S_1 \cap S_2 = \{4, 5, 6\}$.

A.3 Privacy Preserving Set Union

Frikken's two party HBC set union protocol was presented in Section 5.1.2. A sample implementation of this protocol using the Damgård-jurik cryptosystem is presented in this section.

Assume that P_1 possesses the set $S_1 = \{1, 2, 3, 4\}$ and P_2 possesses the set $S_2 = \{3, 4, 5, 6\}$, with both sets containing elements from the domain $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The **encPoly** and **encEval** methods from the previous section will also be used.

A.3.1 Step 1 and 2

P_1 creates an instance of the cryptosystem:

```
> k := gen(16, 2):
> pk := k[1]:
> sk := k[2]:
```

and computes the polynomial representation of S_1:

```
> S1 := [1,2,3,4]:
> f := 1:
> for i from 1 to 4 by 1 do
>    f := f * (x - S1[i]);
> od:
> f := sort(collect(f,x));
> f := f mod pk[1]^(pk[3]):
```

resulting in the polynomial

$$f := x^4 - 10x^3 + 35x^2 - 50x + 24.$$

P_1 then sends an encryption of f to P_2

```
> F := encPoly(pk, f):
> for i from 0 to ArrayNumElems(F) - 1 by 1 do
>    printf("\t F[%d] = %d\n", i, F[i]);
> od:

  F[0] = 4272130759882883072409449647l4
  F[1] = 141150826148801473275823404462
  F[2] = 134861126169660970983037958768
  F[3] = 416324043912280048286289428729
  F[4] = 58075863898780880626933575706
```

A.3.2 Step 3 and 4

For each $s_i \in S_2$, P_2 chooses a random r and computes the tuple $[(E(pk, f(s) * s * r), E(pk, f(s) * r)]$:

```
> S2 := [3, 4, 5, 6]:
> m1 := [0,0,0,0]:
> m2 := [0,0,0,0]:
> m  := [0,0,0,0]:
> for i from 1 to 4 by 1 do
>    r := RandomTools[Generate](integer(range=0..pk[1])):
>    m1[i] := encEval(pk, F, S2[i])&^(r*S2[i])
         mod pk[1]^(pk[3] + 1):
>    m2[i] := encEval(pk, F, S2[i])&^r mod pk[1]^(pk[3] + 1):
>    m[i] := [ m1[i], m2[i] ]:
>    printf("\t(m1[%d], m2[%d]) = (%d, %d)\n",
              i, i, m1[i], m2[i]);
```

```
> od:

   (m1[1], m2[1]) = (118803941814919847773155023447,
                     285384286748733378780702498661)
   (m1[2], m2[2]) = (284898925435807388167298086289,
                     257266985055935850972077024478)
   (m1[3], m2[3]) = (99711422078856642150483O120,
                     113487459739533068126415066208)
   (m1[4], m2[4]) = (149847555967236497758747293549,
                     252757767817564764672766417940)
```

P_2 then randomly permutes each tuple and sends the result to P_1:

```
perm := combinat[permute](m):
> m := perm[rand() mod 4!]:
> for i from 1 to 4 by 1 do
>    m1[i] := m[i][1];
>    m2[i] := m[i][2];
>    printf("\t(m1[%d], m2[%d]) = (%d, %d)\n",
              i, i, m1[i], m2[i]);
> od:

   (m1[1], m2[1]) = (284898925435807388167298086289,
                     257266985055935850972077024478)
   (m1[2], m2[2]) = (99711422078856642150483O120,
                     113487459739533068126415066208)
   (m1[3], m2[3]) = (149847555967236497758747293549,
                     252757767817564764672766417940)
   (m1[4], m2[4]) = (118803941814919847773155023447,
                     285384286748733378780702498661)
```

A.3.3 Step 5

P_1 decrypts each message to learn

```
> for i from 1 to 4 by 1 do
>    m1[i] := dec(sk, m1[i]):
>    m2[i] := dec(sk, m2[i]):
>    if m2[i] <> 0 then
>       printf("\tm1[%d] / m2[%d] = %d\n",
               i, i, m1[i]/m2[i] mod pk[1]^(pk[3]));
>    else
>       printf("\tm1[%d] = m2[%d] = %d\n", i, i, m1[i]);
>    fi:
```

```
> od:

  m1[1] = m2[1] = 0
  m1[2] / m2[2] = 5
  m1[3] / m2[3] = 6
  m1[4] = m2[4] = 0
```

and concludes that $S_1 \cup S_2 = S_1 \cup \{5, 6\}$. Thus, P_1 learns that $\{5, 6\} \in S_1 \cup S_2$, but does not learn that $\{3, 4\} \in S_2$ as well.