New Designs of Encryption Schemes

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Computer Science)

by

Mona Sergi

December 2012

Abstract

Malleability of an encryption concerns the ability of users to compute the encryption of $f(m)$ from the encryption of m for an arbitrary message m and a function $f \in \mathcal{F}$. In this context, we say that the encryption scheme is non-malleable if $\mathcal{F}$ is an empty set, and we say the encryption scheme is fully malleable or fully homomorphic if $\mathcal{F}$ includes any arbitrary function. In some applications the encryption scheme should be non-malleable to preserve the privacy of data while in others malleability of the encryption facilitates the functionality or efficiency of the application.

In this thesis we bring forth several general constructions of non-malleable encryption schemes enjoying different levels of security. We also design a multi-party computation protocol that employs a fully homomorphic encryption scheme to optimize the communication cost.

Approval Sheet

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

Mona Sergi

This dissertation has been read and approved by the Examining Committee:

---

abhi shelat, Advisor

Gabriel Robins, Committee Chair

Steven Myers

Westley R. Weimer

Andrei S. Rapinchuk

Accepted for the School of Engineering and Applied Science:

James H. Aylor, Dean, School of Engineering and Applied Science

December 2012

to my husband
James Armontrout

Acknowledgements

No one succeeds in putting together a successful thesis on their own, and my case is no exception. First I would like to thank my husband James for the happiness, peace, and love he has brought to our life together and for his unflinching support throughout. I would like to thank my parents and my Uncle Younes for always believing in and encouraging me through the inevitable highs and lows of a doctoral thesis. I must also thank my labmates Chi-hao, Ben, Robbie, and Nathan not only for serving as sounding boards for my ideas and as extra eyes for review but for the atmosphere of camaraderie they helped foster. I would like to thank my committee members Gabriel Robins, Steven Myers, Westley Weimer, Andrei Rapinchuk, and abhi shelat for their time and for their valuable insights in taking my thesis from a proposal to a finished project. I also wish to thank Steve Myers who was an invaluable collaborator on several of my works and who at times felt like a second advisor to me. Most importantly I would like to thank my advisor abhi shelat, whose guidance led me to be the computer scientist that I am today and whose ongoing support allowed my project to become reality. Finally, I wish to thank all of the dear friends, family and colleagues not named above who have made my years in Charlottesville such wonderful ones and my doctoral thesis a successful one; whether or not your name appears on this page know that you remain in my thoughts. To all of you I will be forever grateful.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

THis thesis aims to study a quintessential cryptographic primitive for which many basic questions remain. This primitive is encryption which is the most widely used and oldest notion in cryptography. Encryption schemes are used to ensure secure communication between parties where the means of communication might be controlled by an untrusted party or *the adversary*. Encryption is the process of transforming a piece of data that needs to be hidden or the plaintext into a bit string or ciphertext that sounds meaningless unless one has access to data called the *key*. The key can be used later to *decrypt* the ciphertexts. We first discuss the importance of encryption schemes in Section 1.1. Then in Section 1.2, we discuss the security notions regarding encryption schemes and pose four key research questions along with an overview of our approach to answering each in Sections 1.2.2 to 1.2.4 .

1.1 Encryption in Practice

Encryption has become a day to day necessity for preserving privacy in systems ranging from small companies to international businesses, and even to financial institututions such as banks and stock markets. In fact, a report report by the Computer Security Institute [1] shows that the companies surveyed encrypted 66.2% of data in transit and 59.8% of data in storage. Encryptions schemes are essential to the operation of bank automatic teller machines, electronic commerce, Bluetooth devices, digital rights management systems (that prevent unauthorized use of copy righted digital material), and much more.

Data encryption increases time, space and cost overheads in any or all phases of design, implementation and maintenance of a system, and one might reasonably ask if it is worth the added effort. A brief review of the potential consequences of data breaches, however, makes clear that the answer is yes. According to a report published by Privacy Rights Clearinghouse [2], in the United States during 2011 alone 535 data breach incidents were perpetrated resulting in the theft of over 30.4 million sensitive consumer records such as debit card, credit card and social security numbers. This brings the total reported records breached in the United States since 2005 to the alarming number of 543 million. Unfortunately, this is not even the total number of data breach incidents. Beth Givens, the director of Privacy Rights Clearinghouse, says:

> "This is a conservative number. We generally learn about breaches that garner media attention. Unfortunately, many do not. And, because many states do not require companies to report data breaches to a central clearinghouse, data breaches occur that we never hear about. Our Chronology is only a sampling."

The costs of a data breach can be staggering, amounting to millions or even billions of dollars. For example, in April 2011 Sony experienced a data breach within their PlayStation Network leaking the name, address and possibly credit card numbers of 77 million users and costing the company an estimate of 13 billion dollars [3]. Unfortunately, malicious attackers are commonly the cause of data breach incidents. According to 2010 Ponemon Institute benchmark study [4] malicious or criminal attacks are the most expensive cause of data breaches and account for almost one third of all incidents.

Cyber security, which commonly deploys encryption as a tool, plays a major role in today's economy and in both private and public security. President Obama has acknowledged that "the cyber threat is one of the most serious economic and national security challenges we face as a nation," and that "America's economic prosperity in the 21st century will depend on cyber security" [5].

1.2 Encryption in Theory

Encryption schemes can be divided into two broad categories: Private (or Symmetric) Key Encryption and Public (or Asymmetric) Key Encryption. As their names suggest, in the former case the

same key is used to either encrypt or decrypt data, while in the latter case the key for encryption (called public key) is different from the key for decryption (called private key). In general, Private Key Encryption is more efficient than Public Key Encryption. However, in many applications it is not feasible to give away the key that can decrypt ciphertexts. For example, consider a simple auction application where users encrypt their bids and broadcast them to the public. After everyone broadcasts the encryption of their bids, a trusted authority decrypts the bids and announces the winner. Clearly the bidders' privacy requires that their bids stay secret to the public. Hence the encryption and decryption key may not be the same and we need to use a Public Key Encryption Scheme. In this thesis we only focus on Public Key Encryption, and unless otherwise mentioned, we refer to Public Key Encryption Schemes as Encryption Schemes.

The main goal of encryption is to guarantee the privacy of data meaning that a malicious user (or the adversary) cannot learn any information about encrypted data by analyzing its corresponding ciphertext (called the challenge ciphertext). The most basic security guarantee that encryption may provide is semantic security (or CPA security) first introduced by Goldwasser and Micali in [6]. Intuitively, this form of security states that whatever the advisory may learn by analyzing the challenge ciphertext can be learned without access to the challenge ciphertext which roughly means that ciphertexts do not leak any unwanted information.

However, an encryption scheme that satisfies CPA security might not necessarily preserve the privacy of data in some applications because of the (possibly) unrealistic restrictions of the adversary in the CPA security definition which do not necessarily match the restrictions of the malicious users in real world applications. For example, the adversary does not have access to the decryption oracle either before or after seeing the challenge ciphertext in the CPA security definition while a malicious user might have access to the decryption oracle before knowing the challenge ciphertext. In this case if the encryption scheme stays secure, we say the encryption scheme satisfies a stronger notion of security called CCA1 security which was first introduced and defined by Naor and Yung in [7] and intuitively implies that the decryption oracle does not leak the secret key by answering decryption queries. The encryption schemes in [8] are a few examples of

CCA1 encryption schemes.

Unfortunately, even CCA1 security does not prevent the adversary from "cheating" in many applications. For example in our auction application, the adversary might meaningfully maul everyone else's ciphertexts to produce an encryption of one bid over the maximum bid of all other participants. Neither CPA nor CCA1 security can prevent such cheating behavior because in both the adversary wins only by guessing the encrypted data correctly rather than mauling ciphertexts. If an encryption scheme is secure against this attack, we say the encryption scheme is non-malleable. The seminal work of Dolev, Dwork, and Naor [9] introduced the area of non-malleable cryptographic primitives. In short, non-malleability corresponds to inability of the adversary to maul a ciphertext into a related ciphertext (i.e. a ciphertext that encrypts a related message).

In the following sections we discuss non-malleability in greater detail and pose four key research question along with our approach to answering them. We define a parallel query to be an unbounded number of queries all submitted to the decryption oracle at once (the adversary is never allowed to ask the challenge ciphertext from the decryption oracle).

1.2.1 NM-CPA Encryption Schemes

In this type of security definition, the adversary has no access to the decryption oracle prior to receiving the challenge ciphertext. However, after knowing the challenge ciphertext the adversary can ask one parallel query from the decryption oracle. The adversary wins if she can guess the encrypted message in the challenge ciphertext correctly after asking the parallel query. This definition of non-malleability, called NM-CPA, is a strengthened form of the original definition of non-malleability (that was presented in [9]) offered by Pass, shelat and Vaikuntanathan [10]. They also presented a construction from CPA to non-malleable CPA using non-blackbox use of the original encryption scheme. Later, Choi et al. [11] showed how non-malleable CPA encryption can be constructed from standard versions of CPA secure encryption in a black-box manner.

1.2.2 NM-CCA1 Encryption Schemes

This security notion is a combination of CCA1 and NM-CPA security. In more detail, the adversary has unconditional access to the decryption oracle before seeing the challenge ciphertext for an unbounded number of times just like in the CCA1 security definition but can only ask one parallel query after knowing the challenge ciphertext just like in the NM-CPA security definition. Although there exist provably CCA1 encryption schemes in the literature (e.g. [8]), determining whether there exists a NM-CCA1 that is not CCA2 secure (a stronger notion of security) remains an open problem. This blemish on our understanding of the theory of encryption has remained despite multiple advances including many novel techniques for constructing encryption schemes.

> **RQ 1.** As with the case for CPA security, can a CCA1 encryption scheme be transformed into an NM-CCA1 encryption scheme?

Although we do not fully resolve this quesiton, in Chapter 3 we identify a meaningful subset of CCA1 schemes that imply NM-CCA1 security. This subset of CCA1 encryption schemes are plaintext aware under multiple keys and weakly simulatable encryption schemes (we will formally define these concepts later). Intuitively, an encryption scheme is plaintext aware (called **sPA1** in [8]) if the "only" way that a ppt adversary can produce a valid ciphertext is to apply the (randomized) encryption algorithm to the public key and a message [8]. Notice that this definition does not imply non-malleability since there is no guarantee of what an adversary can do *when given a valid ciphertext*. In fact, both encryption schemes from [8] are multiplicatively homomorphic. The weakly simulatable property in our construction is required for technical reasons and roughly corresponds to the ability to to sample ciphertexts and pseudo-ciphertexts with random coins used to generate them.

> **(Informal Theorem)** There exists a black-box construction of NM-CCA1 encryption from plaintext awareness and weak simulatability.

Refer to Chapter 3 for more details.

Next we consider designing encryption schemes which security is strictly stronger than CCA1 security but weaker than CCA2 security (in the CCA2 security definition, the adversary has

unbounded access to the decryption oracle both before and after seeing the challenge ciphertext). Note that there exist encryption schemes that satisfy security notions that "sit between" standard notions. One such example from Cramer et al. [12] consists of a black-box construction of a q-bounded CCA2 encryption scheme which is not NM-CPA, but which satisfies a stronger security notion than CPA. In particular, as a generalization of NM-CPA, Matsuda and Matsuura [13] put forth the challenge of constructing encryption schemes that can handle more than one parallel query after revealing the challenge ciphertext. They write:

> "Since any (unbounded) CCA secure public key encryption construction from CPA secure ones must first be secure against adversaries who make two or more parallel decryption queries, we believe that overcoming this barrier of two parallel queries is worth tackling."

In this spirit, we define an extension over NM-CCA1, cNM-CCA1, that is defined identically to NM-CCA1 except that the adversary can make c adaptive parallel decryption queries after seeing the challenge ciphertext, where each parallel decryption query can request that a polynomial number of ciphertexts be decrypted (excluding the challenge ciphertext). Then we show how to construct a cNM-CCA1 secure encryption scheme for an arbitrary constant c.

> **(Informal Theorem)** There exists a black-box construction of cNM-CCA1 encryption in which the adversary may ask constant number of parallel queries after receiving the challenge ciphertext from plaintext awareness and weak simulatability assumptions.

For more details, see Chapter 3.

1.2.3 NM-CCA2 Encryption Schemes (or CCA2 Encryption Schemes)

CCA2 security is stronger than all other mentioned security notions. As mentioned earlier, in this type of security definition the adversary has access to the decryption oracle both before and after seeing the challenge ciphertext for an unbounded number of times. As before, the adversary is not allowed to ask the challenge ciphertext from the decryption oracle. Despite two decades of rigorous study it remains unclear how to construct a non-malleable encryption scheme (CCA2 secure encryption scheme) from any semantically secure encryption scheme. In this regard, partial

progress has been made by Gertner, Malkin and Myers [14] who showed there are no black-box constructions in which the decryption algorithm of the proposed CCA2 encryption scheme does not query the encryption algorithm of the semantically secure one.

The current state-of-the-art of construction of CCA2 encryption schemes from general assumptions is that of Wee [15] based on a primitive called the extractable hash proof system which is a special kind of non-interactive zero-knowledge proof of knowledge system. It is the first practical CCA2 encryption scheme from general assumptions that can be instantiated from both Computational Diffie-Hellman (CDH) and the factoring assumption. Given a hash proof system, the CCA2 encryption scheme can be constructed in a straightforward way. However the extractable hash proof system is specifically tailored to fit in their framework with quite a complicated and non-intuitive definition.

> **RQ 2.** What are the weakest encryptions from which we can achieve CCA2 security in a general and black-box manner?

In this thesis we take a step towards answering this question by building a provably CCA2 secure encryption scheme from encryptions that do not necessarily satisfy even CPA security. More specifically, the basic needed encryption scheme in our constructions should satisfy two properties: the encryption should be secure under a one-way plaintext-checking attack (i.e. OW-PCA-secure) introduced in [16] and the scheme must support a "key malleability" property. We will formally define each of these notions later, but intuitively an encryption scheme is OW-PCA secure if given the public key and an encryption of a random message m, no ppt adversary can guess m correctly except with a negligible probability even with access to an oracle that takes a ciphertext and a message and verifies whether the ciphertext decrypts to the given message or not. This notion is seemingly weak: the oracle only provides "yes/no" answers about whether a decryption is correct, and the adversary must recover the entire message, and not just any single predicate as per semantic security. The *key malleability* property is required for technical reasons and roughly corresponds to the ability to finding keys that provably leak the same bits of plaintexts as a given key and transforming ciphertexts between the two sets of keys.

We use an encryption scheme that is OW-PCA secure and key malleable to construct an encryption scheme that is one-bit CCA2 secure (i.e. the ciphertext encrypts a single bit).Since Myers and shelat [17] showed that one-bit CCA2 security implies many-bit CCA2 security, we conclude that many-bit CCA2 security may be achieved using an encryption scheme that is only OW-PCA secure and key malleable.

> **(Informal Theorem)** CCA2 security can be achieved from any OW-PCA secure encryption scheme that is key malleable.

See Chapter 4 for more details.

1.2.4 Fully Homomorphic Encryption Schemes

Finally, we consider the opposite question. If an encryption scheme is the opposite of non-malleable, i.e fully homomorphic (an encryption scheme that allows the user to do both addition and multiplication on the ciphertexts), what novel applications become possible? Specifically we focused on the application of fully homomorphic encryption schemes in the domain of multiparty computation protocols (MPC). In an MPC protocol several parties gather to evaluate a public function on the set of private data that each party holds with the goal being that each party would learn only her input and the output of the evaluation.

> **RQ 3.** Can fully homomorphic encryption schemes improve the efficiency of multiparty computation protocols?

We show that using a fully homomorphic encryption scheme helps generate more efficient designs of secure multiparty computation (MPC) protocols. Encryption schemes are commonly used in the design of the MPC protocols to allow for secrecy of the data. There are overheads in such protocols such as computation overhead, communication overhead, security assumptions or round complexity, and our goal is to minimize each of these overheads in our design. Unfortunately there is no single design that optimizes the overhead of MPC in all mentioned areas, but there are designs that are optimal in reducing one or some of the mentioned overheads. Deciding which MPC protocol to use depends on the application. In this thesis we propose a black-box MPC protocol to optimize the communication complexity using fully homomorphic encryption schemes.

(Informal Theorem) Using fully homomorphic encryption schemes, there exists black-box multiparty computation protocols to evaluate an arbitrary function f with communication complexity independent of the size of f.

See Chapter 5 for more details.

Chapter 2

Basic Definitions and Notations

2.1 Notation

We say a function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if for all polynomials p and all sufficiently large k: $\epsilon(k) \leq 1/p(k)$. Unless otherwise mentioned, $\epsilon(\cdot)$ is a negligible function throughout this thesis. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a finite set X, we denote by $x \leftarrow X$ the experiment of choosing an element of X according to the uniform distribution over X. For a distribution $\mathcal{D}$ over a set X, we denote by $x \leftarrow \mathcal{D}$ the experiment of choosing an element of X according to the distribution $\mathcal{D}$. Given finite distributions D_0 and D_1, we denote by $D_0 \equiv D_1$ the fact that the distributions are equal. In X and Y are random variables with clearly defined underlying experiments, that have a shared finite support over their range, we abuse notation and use $X \equiv Y$ to denote that they have the same implied distribution. Given two families of distributions, or random variables $X = \{X_i\}_{i\in\mathbb{N}}$ and $Y = \{Y_i\}_{i\in\mathbb{N}}$ we denote by $X \approx_c Y$ the fact that X and Y are computationally indistinguishable. Similarly, $X \approx_s Y$ denotes that they are statistically indistinguishable.

2.2 Adversary Properties

Definition 2.2.1. *(ppt) A machine is* probabilistic polynomial time (ppt) *if it has access to a random tape that produces random bits upon access and if its running time is bounded by some polynomial in the input size (or often a polynomial in a security parameter).*

Definition 2.2.2. *(Non-uniform ppt Machine [18]). A* non-uniform ppt machine $\mathcal{A}$ *is a sequence of probabilistic machines* $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ *for which there exists a polynomial* d *such that the description size of* $|\mathcal{A}_i| < d(i)$ *and the running time of* $\mathcal{A}_i$ *is also less than* $d(i)$*. We write* $\mathcal{A}(x)$ *to denote the distribution obtained by running* $\mathcal{A}_{|x|}(x)$*.*

2.3 One-Way Functions

Definition 2.3.1. *(Collection of One-Way Functions [18]). A* collection of one-way functions *is a family* $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in I}$ *satisfying the following conditions:*

1. *It is easy to sample a function, i.e. there exists a ppt Gen such that Gen*(1^k) *outputs some* $i \in I$*.*

2. *It is easy to sample a given domain, i.e. there exists a ppt that on input* i *returns a uniformly random element of* $\mathcal{D}_i$*.*

3. *It is easy to evaluate, i.e. there exists a ppt that on input* $i, x \in \mathcal{D}_i$ *computes* $f_i(x)$*.*

4. *It is hard to invert, i.e. for any ppt* $\mathcal{A}$ *there exists a negligible function* $\epsilon(.)$ *such that for all sufficiently large* $k \in \mathbb{N}$

$$\Pr[i \leftarrow \mathit{Gen}(1^k); x \leftarrow \mathcal{D}_i; y \leftarrow f_i(x); x' \leftarrow \mathcal{A}(1^k, i, y) : f_i(x') = y] \leq \epsilon(k)$$

Definition 2.3.2. *(Collection of One-Way Permutations [18]). A collection* $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in I}$ *is a collection of* one-way permutations *if* $\mathcal{F}$ *is a collection of* one-way funcitons *and for all* $i \in I$*, we have that* f_i *is a permutation.*

Definition 2.3.3. *(Collection of Trapdoor Permutations [18]). A* collection of trapdoor permutations *is a family* $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ *satisfying the following properties:*

1. $\forall i \in \mathcal{I}$*,* f_i *is a permutation.*

2. *It is easy to sample a function, i.e. there exists a ppt Gen such that Gen*(1^k) *outputs some* $i \in \mathcal{I}$ *and the trapdoor information* t*.*

3. *It is easy to sample a given domain, i.e. there exists a ppt that on input $i \in \mathcal{I}$ returns a uniformly random element of $\mathcal{D}_i$.*

4. *It is easy to evaluate, i.e. there exists a ppt that on input $i \in \mathcal{I}, x \in \mathcal{D}_i$ computes $f_i(x)$.*

5. *f_i is hard to invert, i.e. for any ppt $\mathcal{A}$ there exists a negligible function $\epsilon(.)$ such that for all sufficiently large $k \in \mathbb{N}$*

$$\Pr[(i,t) \leftarrow \mathit{Gen}(1^k); x \leftarrow \mathcal{D}_i; y \leftarrow f_i(x); x' \leftarrow \mathcal{A}(1^k, i, y) : f_i(x') = y] \leq \epsilon(k)$$

6. *f_i is easy to invert with trapdoor information, i.e. there exists a ppt machine that given input (i,t) from Gen and $y \in \mathcal{R}$, computes $f_i^{-1}(y)$.*

Definition 2.3.4. *(Hard-core Predicate [18]). A predicate $h : \{0,1\}^* \to \{0,1\}$ is a hard-core predicate for $f(x)$ if h is efficiently computable given x, and for all nonuniform ppt adversaries $\mathcal{A}$, there exists a negligible function $\epsilon(.)$ so that for all sufficiently large $k \in \mathbb{N}$*

$$\Pr[x \leftarrow \{0,1\}^k : \mathcal{A}(1^k, f(x)) = h(x)] \leq 1/2 + \epsilon(k)$$

2.4 Indistinguishability

Definition 2.4.1. *(Computational Indistinguishability) We say two distribution ensembles $\{C\}_{k\in\mathbb{N}}$ and $\{D\}_{k\in\mathbb{N}}$ indexed by the security parameter k are* computationally indistinguishable *if for any non-uniform ppt adversary $\mathcal{A}$, the function $\delta_{\mathcal{A}}(.)$, defined as follows, is a negligible function:*

$$\delta_{\mathcal{A}}(k) = |\Pr_{\alpha \leftarrow C_k}[\mathcal{A}(\alpha) = 1] - \Pr_{\alpha \leftarrow D_k}[\mathcal{A}(\alpha) = 1]|$$

Definition 2.4.2. *(Statistical Indistinguishability) We say two distribution ensembles $\{C\}_{k\in\mathbb{N}}$ and $\{D\}_{k\in\mathbb{N}}$ indexed by the security parameter k are* statistically indistinguishable *if for any non-uniform (not necessarily polynomial time) adversary $\mathcal{A}$, the function $\delta_{\mathcal{A}}(.)$, defined as follows, is a negligible function:*

$$\delta_{\mathcal{A}}(k) = |\Pr_{\alpha \leftarrow C_k}[\mathcal{A}(\alpha) = 1] - \Pr_{\alpha \leftarrow D_k}[\mathcal{A}(\alpha) = 1]|$$

2.5 Public Key Encryption Scheme

Definition 2.5.1. *(Public Key Encryption Scheme [18]). A triple* $(\textbf{\textit{Gen}}, \textbf{\textit{Enc}}, \textbf{\textit{Dec}})$ *is a public key encryption scheme if*

1. $(PK, SK) \leftarrow \textbf{\textit{Gen}}(1^k)$ *is a ppt algorithm that produces a key pair PK (called the public key) and SK (called the secret key)*

2. $c \leftarrow \textbf{\textit{Enc}}_{PK}(m)$ *is a ppt algorithm that given the public key PK and the message* $m \in \{0,1\}^k$ *produces a ciphertext* c

3. $m \leftarrow \textbf{\textit{Dec}}_{SK}(c)$ *is a ppt algorithm that produces a message* $m \in \{0,1\}^k$ *given a ciphertext* c *and the secret key SK*

4. *There exists a polynomial-time algorithm* $\mathcal{M}$ *that on input* 1^k *and* i *outputs the* i^{th} *message of length* k *(if such a message exists)*

5. $\forall k, m \in \{0,1\}^k$*:*

$$\Pr[(PK, SK) \leftarrow \textbf{\textit{Gen}}(1^k) : \textbf{\textit{Dec}}_{SK}(\textbf{\textit{Enc}}_{PK}(m)) = m] = 1$$

The decryption algorithm produces a special symbol $\perp$ when the input ciphertext is "undecipherable". We define the security property for public-key encryption in Section 2.6.

2.6 CPA/CCA1/CCA2 Security

We recall the definitions for chosen plaintext attack (CPA), lunch time chosen ciphertext attack (CCA1), and chosen ciphertext attack (CCA2) that originate from [6], [7] and [19] respectively. We use the notation from [20].

Definition 2.6.1. *(**AAA** Security) For **AAA** $\in \{\mathsf{CPA}, \mathsf{CCA1}, \mathsf{CCA2}\}$, **E** is said to be **AAA** secure if for $k \in \mathbb{N}$, $\boldsymbol{Adv}^{\boldsymbol{AAA}}(\boldsymbol{E}, \boldsymbol{X}, k)$ is negligible for every polynomial-time **AAA**-adversary **X** where*

$$\boldsymbol{Adv}^{\boldsymbol{AAA}}(\boldsymbol{E}, \boldsymbol{X}, k) = \Pr[\boldsymbol{AAA}_1(\boldsymbol{E}, \boldsymbol{X}, k) = 1] - \Pr[\boldsymbol{AAA}_0(\boldsymbol{E}, \boldsymbol{X}, k) = 1]$$

*where **AAA** experiment is defined as follows:*

$\mathbf{AAA}_b(\mathbf{E}, \mathbf{X}, k)$
1: $(\mathrm{PK}, \mathrm{SK}) \xleftarrow{\$} \mathbf{Gen}(1^k)$
2: $(M_0, M_1, St) \xleftarrow{\$} \mathbf{X}^{O_1(.)}(find, \mathrm{PK})$
3: $C \xleftarrow{\$} \mathbf{Enc}_{\mathrm{PK}}(M_b)$
4: $d \leftarrow \mathbf{X}^{O_2(.)}(guess, C)$
5: Return d

In the case of CPA *security, $O_1(.) = \lambda$ and $O_2(.) = \lambda$, in the case of* CCA1 *security, $O_1(.) = \boldsymbol{Dec}_{SK}(.)$ and $O_2(.) = \lambda$, and in case of* CCA2 *security $O_1(.) = \boldsymbol{Dec}_{SK}(.)$ and $O_2(.) = \boldsymbol{Dec}_{SK}(.)$ with the difference that $O_2(.)$ would return $\perp$ if asked on the challenge ciphertext.*

2.7 Strong One-Time Signature Scheme

A strong one-time signature scheme is defined as follows:

Definition 2.7.1. *(Strong One-Time Signature Scheme [21]) A signature scheme $\Sigma = (\mathsf{GenKey}, \mathsf{Sign}, \mathsf{Verify})$ is* a strong one-time signature scheme *if the success probability of any ppt adversary $\mathcal{A}$ in the following game is negligible in the security parameter k:*

1. $\mathsf{GenKey}(1^k)$ *outputs $(vkSig, skSig)$ and the adversary is given 1^k and vkSig,*
2. $\mathcal{A}(1^k, vkSig)$ *may do one of the following:*
   - *$\mathcal{A}$ may output a pair (m^*, σ^*) and halt. In this case (m, σ) are undefined.*
   - *$\mathcal{A}$ may output a message m, and is then given in return $\sigma \leftarrow \mathsf{Sign}_{skSig}(m)$. Following this, $\mathcal{A}$ outputs (m^*, σ^*).*

We say the adversary succeeds if $\mathsf{Verify}_{vkSig}(m^*, \sigma^*) = 1$ *but* $(m^*, \sigma^*) \neq (m, \sigma)$ *(assuming* (m, σ) *are defined). We stress that the adversary may succeed even if* $m^* = m$.

Note that public key encryption implies strong one-time signature. This follows by combining the observations that public key encryption implies one-way functions, one-way functions imply universal one-way hash functions [22], and universal one-way hash functions imply strong one-time signature schemes [23, 24]. Thus, a strong one-time signature can be constructed given any encryption scheme. In this thesis, whenever we need to assume the existence of both encryption schemes and strong one-time signature schemes we only mention assuming the former since the former implies the latter.

Chapter 3

Non-Malleable CCA1 Security and Beyond

3.1 Introduction

The standard security definition of an encryption scheme does not prevent an adversary who observes an encryption of the message m from producing an encryption of the message $f(m)$ for some function f (even though the value m remains private). The seminal work of Dolev, Dwork, and Naor [9] addressed this security issue by introducing the area of non-malleable cryptographic primitives such as encryption schemes, commitment schemes, and zero-knowledge. Later, Pass, shelat and Vaikuntanathan [10] strengthened the DDN definition and presented a construction from CPA to non-malleable CPA using non-blackbox use of the original encryption scheme. There have been many follow-up works that propose more efficient constructions of non-malleable primitives. A notable achievement in this line of research has been the construction of non-malleable primitives using only black-box access to the standard version of the same primitive [11, 25, 15]. In particular, [11] show how non-malleable CPA encryption can be constructed from standard versions of encryption in a black-box manner.

However, the question of whether an NM-CCA1 encryption scheme can be constructed from a CCA1 encryption scheme has remained open. This blemish on our understanding of the theory of encryption has remained despite multiple advances including many novel techniques for constructing encryption schemes. In this work, we present a black-box construction of an NM-CCA1 encryption scheme for a subset of CCA1 encryption schemes, namely those which are also plaintext

aware under multiple keys and weakly simulatable (we will formally define these concepts later). Intuitively, an encryption scheme is plaintext aware (called **sPA1** in [8]) if the only way that a ppt adversary can produce a valid ciphertext is to apply the (randomized) encryption algorithm to the public key and a message [8]. Notice that this definition does not imply non-malleability since there is no guarantee of what an adversary can do *when given a valid ciphertext*. In fact, both encryption schemes from [8] are multiplicatively homomorphic. The weakly simulatable property in our construction is required for technical reasons and roughly corresponds to the ability to to sample ciphertexts and pseudo-ciphertexts with random coins used to generate them.

Note that there exist encryption schemes that satisfy security notions that "sit between" standard notions. One such example from Cramer et al. [12] consists of a black-box construction of a q-bounded CCA2 encryption scheme which is not NM-CPA, but which satisfies a stronger security notion than CPA. In particular, as a generalization of NM-CPA, Matsuda and Matsuura [13] put forth the challenge of constructing encryption schemes that can handle more than one parallel query after revealing the challenge ciphertext. They write:

> "Since any (unbounded) CCA secure PKE construction from IND-CPA secure ones must first be secure against adversaries who make two or more parallel decryption queries, we believe that overcoming this barrier of *two parallel queries* is worth tackling."

In this spirit, we define an extension over NM-CCA1, cNM-CCA1, that is defined identically to NM-CCA1 except that the adversary can make c adaptive parallel decryption queries after seeing the challenge ciphertext, where each parallel decryption query can request that a polynomial number of ciphertexts be decrypted (excluding the challenge ciphertext). (Note that NM-CCA1 is cNM-CCA1 where the parameter c is set to be one.) Then we show how to construct a cNM-CCA1 secure encryption scheme for an arbitrary constant c. Unfortunately, the size of the ciphertext in a cNM-CCA1 encryption scheme is polynomially bigger than the size of the ciphertext in a (c-1)NM-CCA1 encryption scheme and thus the parameter c must be a constant to obtain an efficient construction.

About Knowledge Extraction Assumptions Our constructions rely on encryption schemes that are plaintext aware (**sPA1**$_\ell$) in the multi-key setup and are weakly simulatable. In Theorem 3.2.9, we show that such encryption schemes exist under a suitable extension of the Diffie-Hellman Knowledge (DHK) assumption that was originally proposed by Damgård, and modified to permit interactive extractors by Bellare and Palacio [8]. Dent [26] has since shown that it is secure in the generic group model. We understand that there are some critics of the DHK assumption, due to its strength and the fact that it is not efficiently falsifiable. However, it is not our goal to argue whether or not it is an assumption which should be used in deployable systems. Instead we note it is seemingly a weaker assumption than the Random Oracle model (which is known to be incorrect in full generality, cf. [27]), under which it is relatively easy to show that simple CPA secure encryption schemes imply CCA2 secure ones. In contradistinction, there are no security definitions that seem weaker or incomparable to NM-CCA1 that are known to imply schemes which are NM-CCA1. Similarly, the gap between NM-CCA1 and CCA2 is poorly understood.

Techniques Similar to the nested encryption construction in [28], both our NM-CCA1 and cNM-CCA1 constructions are based on the notion of double encryption. We first encrypt the message under one key (we refer to this ciphertext as the "inner layer"), and encrypt the resulting inner layer ciphertext repetitively under an additional k keys, where k is the security parameter (we refer to these k keys as the "outer keys", and the ciphertexts they produce as the "outer layer"). During decryption, all the outer layer ciphertexts are decrypted, and it is verified they all encode the same inner layer value. This is combined with the well studied notion of non-duplicatable set selection (in this case of public-keys used to encrypt the outer-layer encryptions), such that anyone attempting to maul a ciphertext has to perform their own independent outer layer encryption. Intuitively, anyone that can encrypt to a consistent outer layer encryption under a new key must have knowledge of the underlying inner-layer, and thus a valid ciphertext is not mauled.

On a more technical level, there are several challenges that need to be overcome. The traditional technical difficulty in proving weaker public-key encryption security notions imply stronger security notions is in showing how to simulate the decryption oracle. When beginning with a **sPA1**$_\ell$-secure

encryption primitive, we can easily simulate the initial decryption oracle in the NM-CCA1 security definition, which is present before the challenge ciphertext is presented, by using the extractor guaranteed by the $\mathbf{sPA1}_\ell$ security definition. However, we cannot simply use the extractor to simulate the decryption oracle after receiving the challenge ciphertext in the NM-CCA1 security experiment. This is because the plaintext aware security does not guarantee that an extractor could decrypt ciphertexts where the underlying randomness is not known to the party that created the ciphertext. Generally, a party that mauls a ciphertext as in the case of non-malleability will not have access to this underlying randomness. To overcome this problem, we make use of a weak notion of simulatability.

To summarize, our contribution is twofold. Firstly, our work shows the first black-box construction of a non-malleable CCA1 encryption scheme in the standard model that is not CCA2 secure. Secondly, for the first time, we show how to construct an encryption scheme that is not CCA2 secure but is secure against an adversary that can ask a bounded number of polynomial-parallel queries after receiving the challenge ciphertext, satisfying a natural extension to the notion of NM-CCA1 security. This might be of independent interest since the development of constructions that satisfy stronger notions than non-malleable CCA1 security but do not satisfy CCA2 security can provide insight in trying to understand the technical difficulties in understanding the larger relationship between CCA1 and CCA2.

3.2 Notations and Definitions

Although non-malleability can be defined for any CPA, CCA1 or CCA2 encryption scheme (we use the standard definition for CPA/CCA1/CCA2 security), we only use and hence only define non-malleability for CCA1 encryption schemes. We use a definition similar to the non-malleability definition for CPA encryption schemes in [10].

Definition 3.2.1 (NM-CCA1)**.** *We say that* $\boldsymbol{E} = (\mathsf{nmg}, \mathsf{nme}, \mathsf{nmd})$ *is non-malleable* CCA1 *secure if for all ppt adversaries and ppt distinguishers* $\mathcal{A}$ *and* $\mathcal{D}$ *respectively and for all polynomials* $p(\cdot)$*,*

we have that $\{\mathsf{NME}_0(\boldsymbol{E}, \mathcal{A}, \mathcal{D}, k, p(k))\}_k \approx_c \{\mathsf{NME}_1(\boldsymbol{E}, \mathcal{A}, \mathcal{D}, k, p(k))\}_k$ *where the experiment* NME *is defined in Figure 3.1.*

$\mathsf{NME}_b(\mathbf{E}, \mathcal{A}, \mathcal{D}, k, p(k))$
1: $(\text{NPK}, \text{NSK}) \leftarrow \mathsf{nmg}(1^k)$
2: $(m_0, m_1, S_1) \leftarrow \mathcal{A}_1^{\mathsf{nmd}_{\text{NSK}}}(\text{NPK})$ s.t. $|m_0| = |m_1|$
3: $y \leftarrow \mathsf{nme}(\text{NPK}, m_b)$
4: $(\vec{c}, S_2) \leftarrow \mathcal{A}_2(y, S_1)$ where $|\vec{c}| = p(k)$
5: Output $\mathcal{D}(\vec{d}, S_2)$ where $d_i \leftarrow \mathsf{nmd}(\text{NSK}, c_i)$ if $c_i \neq y$ and $d_i \leftarrow \perp$ if $c_i = y$

Figure 3.1: THE NME EXPERIMENT USED TO DEFINE NM-CCA1 SECURITY

3.2.1 Weakly Simulatable Encryption Scheme

Dent in [29] introduced the notion of simulatability for an encryption scheme. Intuitively, an encryption scheme is simulatable if no attacker can distinguish valid ciphertexts from some family of pseudo-ciphertexts (which will include both valid encryptions and invalid encryptions). This family of pseudo-ciphertexts must be efficiently and publicly computable (i.e. without access to any private knowledge, say related to the secret key), and somewhat invertible (given a pseudo-ciphertext, one can find a random looking string that generates it). In Dent's definition, the attacker also has access to a decryption oracle to help it distinguish between pseudo-ciphertexts and legitimate ones, but it cannot query the decryption oracle on the challenges that it is trying to distinguish.

For our purposes, consider a restricted notion of simulatability where the attacker is not given access to the decryption oracle. If an encryption scheme satisfies this weaker notion of simulatability, we say it is weakly simulatable.

Definition 3.2.2. ***(Weakly Simulatable Encryption Scheme)*** *An asymmetric encryption scheme* $(\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *is weakly simulatable if there exist two poly-time algorithms* (f, f^{-1})*, where* f *is deterministic and* f^{-1} *is probabilistic, such that for all* $k \in \mathbb{N}$ *there exists the polynomial function* $p(.)$ *where* $l = p(k)$*, we have the following correctness properties:*

1. *f on inputs of public key PK (in the range of **Gen**) and a random string $r \in \{0,1\}^l$, returns elements in $\mathcal{C}$, where $\mathcal{C}$ is the set of all possible "ciphertext"-strings that can be submitted to the decryption oracle (notice that members of $\mathcal{C}$ are both valid and invalid ciphertexts).*

2. *f^{-1} on input of a public key PK (in the range of **Gen**) and an element $C \in \mathcal{C}$, outputs elements of $\{0,1\}^l$.*

3. *$f(PK, f^{-1}(PK, C)) = C$ for all $C \in \mathcal{C}$.*

And the following security properties. No polynomial time attacker $\mathcal{A}$ has probability better than $1/2 + \mu(k)$ of winning in the following experiment, where μ is some negligible function.

1. *The challenger generates a random key pair $(PK, SK) \leftarrow \textbf{\textit{Gen}}(1^k)$, and chooses randomly $b \in \{0,1\}$.*

2. *The attacker $\mathcal{A}$ executes on the input 1^k and the public key PK outputs $m \in \mathcal{M}$. The challenger sends $\mathcal{A}$ the pair $(f^{-1}(PK, c = \textbf{\textit{Enc}}_{PK}(m)), c)$ if $b = 0$, or $(r, f(PK, r))$ for some randomly generated element $r \in \{0,1\}^l$ if $b = 1$. The attacker $\mathcal{A}$ terminates by outputting a guess b' for b.*

   $\mathcal{A}$ wins if $b = b'$ and its advantage is defined in the usual way.

In a scheme where you cannot distinguish legitimate ciphertexts from pseudo-ciphertexts that need not encode actual messages, CPA security is immediate. The converse need not hold, as ciphertexts might be hard to generate, and invalid ciphertexts might be easily distinguishable from illegitimate ones (for example, they might contain a zero-knowledge proof of validity). Notice that the weak simulatability notion is not equivalent to the Invertible Sampling notion introduced in [30] since the plaintext is not needed to compute the random looking string that generates the ciphertext.

Theorem 3.2.3. *If **E** is a weakly simulatable encryption scheme, then **E** is* CPA *secure.*

Proof. Let **E** be weakly simulatable on the pair of turing machines (f, f^{-1}). Let $\mathcal{A}$ be a CPA adversary. Let $\mathcal{B}_{\mathcal{A}}$ be an attacker against the weakly simulatability property of **E** that works as follows:

- The challenger generates a pair of public and secret keys $(\text{PK}, \text{SK}) \leftarrow \textbf{Gen}(1^k)$. The challenger also chooses a random bit $b \in \{0,1\}$.
- The attacker $\mathcal{B}_\mathcal{A}$ gets executed on the public key PK. $\mathcal{B}_\mathcal{A}$ runs $\mathcal{A}_1$ on the input PK until it halts with messages m_0 and m_1 and the state information st.
- The attacker $\mathcal{B}_\mathcal{A}$ chooses a random bit $d \in \{0,1\}$, and queries the challenger on m_d.
- The challenger chooses a random string $r \in \{0,1\}^\ell$ and answers with $(r = f^{-1}(\text{PK}, c), c = \textbf{Enc}_{\text{PK}}(m_d))$ if $b = 0$, and $(r, c = f(\text{PK}, r))$ if $b = 1$.
- $\mathcal{B}_\mathcal{A}$ receives (r, c) from the challenger. $\mathcal{B}_\mathcal{A}$ then runs $\mathcal{A}_2$ on the input c and st until $\mathcal{A}_2$ outputs the bit d' as a guess for d and halts. If $d' = d$, $\mathcal{B}_\mathcal{A}$ outputs $b' = 0$ otherwise it outputs $b' = 1$.

Now we analyze the advantage of $\mathcal{B}_\mathcal{A}$ assuming **E** is not CPA secure. Note that

$$\Pr[d' = d | b = 0] > 1/2 + \epsilon(k) \tag{3.1}$$

for any negligible function $\epsilon(.)$ if **E** is not CPA secure ($\Pr[d' = d|b = 0]$ is the probability that $\mathcal{A}$ guesses the encrypted message correctly when it is given a valid ciphertext). We will show that this inequality is in contradiction with the assumption that **E** is weakly simulatable.

E is weakly simulatable if and only if for some negligible function $\epsilon'(.)$

$$|\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| < \epsilon'(k) \tag{3.2}$$

Note that $b' = 0$ if and only if $d' = d$. Hence $\Pr[b' = 0|b = 0] = \Pr[d = d'|b = 0]$ and $\Pr[b' = 0|b = 1] = \Pr[d = d'|b = 1]$. We have that

$$\Pr[d = d'|b = 1] = 1/2 \tag{3.3}$$

because whenever $b = 1$, the given ciphertext to the adversary is independent of the bit d and hence the advantage of the adversary in guessing the random bit d is exactly $1/2$. Hence

$$|\Pr[b'=0|b=0]-\Pr[b'=0|b=1]| =$$

$$|\Pr[d=d'|b=0]-\Pr[d=d'|b=1]|$$

substituting (3.1), (3.3), $> 1/2+\epsilon(k)-1/2$

$$> \epsilon(k)$$

for any negligible function $\epsilon(.)$ which is in contradiction with the assumption that **E** is weakly simulatable (Equation 3.2). Hence we conclude that **E** is CPA secure.

□

Algorithm 1: DEG

$\mathcal{G}(1^k)$
1: $(p,q,g) \leftarrow G(1^k)$
2: $x_1 \leftarrow \mathbb{Z}_q; X_1 \leftarrow g^{x_1} \mod p$
3: $x_2 \leftarrow \mathbb{Z}_q; X_2 \leftarrow g^{x_2} \mod p$
4: Return $(pk=(p,q,g,X_1,X_2), sk=(p,q,g,x_1,x_2))$

$\mathcal{E}(pk,M)$
1: $y \leftarrow \mathbb{Z}_q; Y \leftarrow g^y \mod p$
2: $W \leftarrow X_1^y; V \leftarrow X_2^y \mod p$
3: $U \leftarrow V \cdot M \mod p$
4: Return $C=(Y,W,U)$

$\mathcal{D}(sk,C)$
1: **if** $W \neq Y^{x_1} \mod p$ **then** Return $\perp$
2: Return $M \leftarrow U \cdot Y^{-x_2} \mod p$

Algorithm 2: CS-Lite

$\mathcal{G}(1^k)$
1: $(p,q,g_1) \leftarrow G(1^k); g_2 \leftarrow G_q \backslash \{1\}$
2: $x_1 \leftarrow \mathbb{Z}_q; x_2 \leftarrow \mathbb{Z}_q; z \leftarrow \mathbb{Z}_q$
3: $X \leftarrow g_1^{x_1}.g_2^{x_2} \mod p; Z \leftarrow g_1^z \mod p$
4: Return $(\text{PK}=(p,q,g_1,g_2,X,Z), \text{SK}=(p,q,g_1,g_2,x_1,x_2,z))$

$\mathcal{E}(pk,M)$
1: $r \leftarrow \mathbb{Z}_q$
2: $R_1 \leftarrow g_1^r \mod p; R_2 \leftarrow g_2^r \mod p$
3: $E \leftarrow Z^r \cdot M \mod p; V \leftarrow X^r \mod p$
4: Return $C=(R_1,R_2,E,V)$

$\mathcal{D}(sk,C)$
1: **if** $V \neq R_1^{x_1} \cdot R_2^{x_2} \mod p$ **then** Return $\perp$
2: Return $M \leftarrow E \cdot R_1^{-z} \mod p$

Table 3.1: THE ENCRYPTION SCHEMES DEG AND CS-LITE

Following the ideas of Dent, in what follows we show how DEG and CS-lite schemes can both be weakly simulatable when instantiated in proper groups. We argue that the Damgård ElGamal (DEG) scheme is weakly simulatable using an argument parallel to that of Dent [29]. We remind the reader that the definition for DEG is given on page 23.

We use the notion of a simulatable group given by Dent [29].

Definition 3.2.4. *(**Simulatable Group**) [29] A Group G is simulatable if there exist two polynomial turing machines (f, f^{-1}) such that:*

- *f is a deterministic turing machine that takes a random element $r \in \{0,1\}^l$ as input, and outputs elements of G.*
- *f^{-1} is a probabilistic turing machine that takes elements of $h \in G$ as input, and outputs elements of $\{0,1\}^l$.*
- *$f(f^{-1}(C)) = C$ for all $h \in G$.*
- *There exists no polynomial time attacker $\mathcal{A}$ that has a non-negligible advantage in winning the following game:*

  1. *The challenger randomly chooses a bit $b \in \{0,1\}$.*
  2. *The attacker $\mathcal{A}$ executes on the input 1^k. The attacker has access to an oracle $\mathcal{O}_f$ that takes no input, generates a random element $r \in \{0,1\}^l$, and returns r if $b = 0$, and $f^{-1}(f(r))$ if $b = 1$. The attacker terminates by outputting a guess b' for b.*

  The attacker wins if $b = b'$ and its advantage is defined in the usual way.

- *There exists no polynomial-time attacker $\mathcal{A}$ that has a non-negligible advantage in winning the following game:*

  1. *The challenger randomly chooses $b \in \{0,1\}$.*
  2. *The attacker $\mathcal{A}$ executes on the input 1^k. The attacker has access to an oracle $\mathcal{O}_f$ that takes no input. If $b = 0$, then the oracle generates a random $r \in \{0,1\}^l$ and returns $f(r)$. Otherwise the oracle generates a random $h \in G$ and returns h. The attacker terminates by outputting a guess b' for b.*

  The attacker wins if $b = b'$ and its advantage is defined in the usual way.

Dent showed that groups in which the DDH assumptions are believed to hold are simulatable.

Lemma 3.2.5. *[29] If q and p are primes such that $p = 2q + 1$, and G is the subgroup of $\mathbb{Z}_p^*$ of order q, then G is simulatable.*

Using this fact we show that DEG is weakly simulatable.

Theorem 3.2.6. *The DEG encryption scheme is weakly simulatable if it is instantiated on a simulatable group G on which the DDH problem is hard.*

Proof. Let (f_G, f_G^{-1}) be the efficiently computable functions that exist by the fact that the group G is a simulatable group. For ease of notation in this proof, we assume all functions get the required public parameters (e.g. the public key) as part of their input. We need to give the two functions (f, f^{-1}) for DEG required by the definition of weakly simulatable. The functions are (f, f^{-1}) where we let $f(x = (x_1, x_2, x_3)) = f_G(x_1), f_G(x_2), f_G(x_3)$, and $f^{-1}(c = (c_1, c_2, c_3)) = f_G^{-1}(c_1), f_G^{-1}(c_2), f_G^{-1}(c_3)$. From this construction and the corresponding properties in the definition of a simulatable group, it is clear that (f, f^{-1}) satisfy the requirements given in the definition of a weakly simulatable encryption scheme.

We now need to argue the final property of a weakly simulatable encryption scheme: no ppt adversary can distinguish between a valid ciphertext and a fake one. To meet this goal, we design a series of games, to show the probability of an adversary being able to distinguish between legitimate ciphertexts, and random outputs of f is negligible. Let W_i be the event that an attacker outputs 1 in Game i.

Let **Game 1** be the portion of the security definition where an attacker is given a randomly generated public-key PK, and it then outputs a message. It is then given a random encryption of that message, and then outputs 1 or 0 (as is explained in step 2 of the weakly simulatable encryption scheme definition).

Let **Game 2** be a game in which the returned ciphertext is made with the following algorithm that always encrypts the message 1 instead of the true encryption algorithm as described in Figure 3.2.

Claim 1. $\Pr[W_1] \approx_c \Pr[W_2]$.

$\mathcal{E}(pk, M)$
1: Randomly select $y \in \mathbb{Z}_q^*$
2: Set $Y = g^y$
3: Set $W = X_1^y$
4: Set $U = X_2^y$
5: Output $(Y, W, U \cdot 1)$

Figure 3.2: THE ENCRYPTION ALGORITHM IN **Game 2**

Proof. Follows from CPA security of the DEG encryption scheme. □

Let **Game 3** be a game in which we modify **Game 2** by modifying encryption so that W is computed as follows:
Randomly select $r' \in \mathbb{Z}_q$ and set $W = g^{r'}$.

Claim 2. $\Pr[W_2] \approx_c \Pr[W_3]$

Proof. If there is any significant difference between $\Pr[W_2]$ and $\Pr[W_3]$ then the attacker can be used to build a DDH distinguisher. In particular, given a tuple $(g, g^{x_1}, g^y, g^{x_1 y})$ or $(g, g^{x_1}, g^y, g^{r'})$. We can choose a random x_2, simulate a PK for the DEG scheme, and use x_2 and the provided information to provide an appropriate encryption, for a perfect simulation of the either **Game 2** or **Game 3**. □

Let **Game 4** be a game in which we modify **Game 3** by modifying encryption so that U is computed as follows:
Randomly select $r' \in \mathbb{Z}_q$ and set $U = g^{r'}$.

Claim 3. $\Pr[W_3] \approx_c \Pr[W_4]$

Proof. The proof this claim parallels that of the previous one. If there is any significant difference between $\Pr[W_2]$ and $\Pr[W_3]$ then the attacker can be used to build a DDH distinguisher. In particular, given a tuple $(g, g^{x_2}, g^y, g^{x_2 y})$ or $(g, g^{x_2}, g^y, g^{r'})$. We can choose a random x_1, simulate a PK for

the DEG scheme, and use x_1 and the provided information to provide an appropriate encryption, for a perfect simulation of the either **Game 3** or **Game 4**. □

In **Game 4** each of the components of the ciphertext is now random group element. In **Game 5** we replace these random group elements with random output from f_G, which due to the simulatable group properties are indistinguishable. Specifically, in **Game 5** we return the elements specified in Figure 3.3 for the encryption algorithm.

$\mathcal{E}(pk, M)$
1: Randomly select $r_1 \in \{0,1\}^l$ and set $Y = f_G(r_1)$
2: Randomly select $r_2 \in \{0,1\}^l$ and set $W = f_G(r_2)$
3: Randomly select $r_3 \in \{0,1\}^l$ and set $U = f_G(r_3)$
4: Output (Y, W, U)

Figure 3.3: THE ENCRYPTION ALGORITHM IN **Game 5**

Claim 4. $\Pr[W_4] \approx_c \Pr[W_5]$

Proof. Follows immediately from simulatable group properties of G and f_G. □

We note that the output of the encryption algorithm in **Game 5** is the same as the output $f(r_1, r_2, r_3)$ for randomly chosen r_1, r_2, r_3. Therefore, since we can combine all the claims to show that $\Pr[W_1] \approx_c \Pr[W_5]$ we conclude that DEG is weakly simulatable.

□

Theorem 3.2.7. *The Cramer-Shoup lite encryption scheme is weakly simulatable if it is instantiated on a simulatable group G on which the DDH problem is hard.*

Proof. Similar to the proof of Theorem 3.2.6. □

3.2.2 Plaintext Awareness For Multiple Key Setup

We present a slight generalization to the definition of **sPA1** by [8] in which multiple keys are permitted to be constructed and given to the ciphertext creator, and the extractor must be able to

sPA1$_\ell(\mathbf{E}, \mathbf{C}, \mathbf{C}^*, k)$

1: Let $R[\mathbf{C}]$, $R[\mathbf{C}^*]$ be randomly chosen bit strings for $\mathbf{C}$ and $\mathbf{C}^*$
2: $((\text{PK}_i, \text{SK}_i))_{i\in[\ell(k)]} \leftarrow \mathbf{Gen}(1^k)$
3: $st \leftarrow \big((\text{PK}_i)_{i\in[\ell(k)]}, R[C]\big)$
4: $\mathbf{C}^{\mathbf{C}^*(st,\cdot)}\big((\text{PK}_i)_{i\in[\ell(k)]}\big)$
5: Let $Q = \{(q_i = (pk_{j_i}, c_i), m_i)\}$ be the set of queries $\mathbf{C}$ made to $\mathbf{C}^*$ until it halted and $\mathbf{C}^*$'s responses to them. Return $\wedge_{i=1}^{|Q|}(m_i = \mathbf{Dec}_{\text{SK}_{j_i}}(c_i))$

Figure 3.4: THE **sPA1**$_\ell$ DEFINITION

decrypt relative to all of the keys. Notice that the **sPA1** definition is a special case of **sPA1**$_\ell$ where $\ell(k) = 1$.

In Figure 3.4, $\mathbf{C}$ is a ciphertext creator, and $\mathbf{C}^*$ is a stateful ppt algorithm called the *extractor* that takes as input the state information st and a ciphertext given by the ciphertext creator $\mathbf{C}$, and will return the decryption of that ciphertext and the updated state st. The state information st is initially set to the public key PK and the adversary $\mathbf{C}$'s random coins. It gets updated by $\mathbf{C}^*$ as $\mathbf{C}^*$ answers each query that the adversary $\mathbf{C}$ submits. The above experiment returns 1 if all the extractor's answers to queries are the true decryption of those queries under SK. Otherwise, the experiment returns 0.

Definition 3.2.8 (sPA1$_\ell$). *Let ℓ be a polynomial. Let $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ be an asymmetric encryption scheme. Let the ciphertext-creator adversary $\boldsymbol{C}$ and the extractor $\boldsymbol{C}^*$ be ppt algorithms. For $k \in \mathbb{N}$, the* **sPA1**-advantage *of $\boldsymbol{C}$ relative to $\boldsymbol{C}^*$ is defined as:*

$$\boldsymbol{Adv}^{\boldsymbol{sPA1}_\ell}(\boldsymbol{E}, \boldsymbol{C}, \boldsymbol{C}^*, k) = \Pr[\boldsymbol{sPA1}_\ell(\boldsymbol{E}, \boldsymbol{C}, \boldsymbol{C}^*, k) = 0]$$

The extractor $\boldsymbol{C}^*$ is a successful **sPA1**$_\ell$-extractor *for the ciphertext-creator adversary $\boldsymbol{C}$ if for all $k \in \mathbb{N}$, the function $\boldsymbol{Adv}^{\boldsymbol{sPA1}_\ell}(\boldsymbol{E}, \boldsymbol{C}, \boldsymbol{C}^*, k)$ is negligible. The encryption scheme $\boldsymbol{E}$ is called* **sPA1**$_\ell$ secure *if for any ppt ciphertext creator there exists a successful $\boldsymbol{sPA1}_\ell$-extractor.*

We argue that Cramer-Shoup Lite (CS-Lite) and Damgard's ElGammal (DEG), described in

Table 3.1, are **sPA1**$_\ell$ secure based on a suitable modification of the Diffie-Hellman Knowledge definition originally proposed by Damgård and modified to permit interactive extractors by Bellare and Palacio [8]. We present this modified version in Figure 3.5.

DHK$_\ell(k)$
1: $(p_i, q_i, g_i \leftarrow G(1^k);\ a_i \leftarrow \mathbb{Z}_{q_i};\ A_i \leftarrow g_i^{a_i} \mod p_i)_{i\in\ell(k)}$
2: Let $R[H]$ and $R[H^*]$ be randomly selected strings for H and H^*
3: $st \leftarrow ((p_i, q_i, g_i, A_i)_{i\in[\ell(k)]}, R[H])$
4: **while** Simulate $H((p_i, q_i, g_i, A_i)_{i\in[\ell(k)]}; R[H])$ **do**
5: **if** H queries (i, B, W) **then**
6: $(b, st) \leftarrow H^*((i, B, W), st; R[H^*])$
7: **if** $W \equiv B^{a_i} \mod p_i$ and $B \not\equiv g_i^b \mod p_i$ **then** Return 1
8: **else** Return b
9: Return 0

Figure 3.5: DHK$_\ell$: AN EXTENSION TO THE DHK DEFINITION

We note that in Figure 3.5, the change requires that the ciphertext creator be able to generate ciphertexts relative to a polynomial number of randomly chosen public keys. It seems reasonable to conjecture that any extractor that could extract exponents with respect to single value $A = g^a$, could do so efficiently for many A_i.

We now argue that DEG is **sPA1**$_\ell$ secure under the DHK$_\ell$ definition.

Theorem 3.2.9. *For any polynomial ℓ, The DEG scheme is **sPA1**$_\ell$ secure under the DHK$_\ell$ assumption.*

Proof. To show that DEG is **sPA1**$_\ell$ secure, we need to show that for any adversary **C** there exists an extractor **C*** that can decrypt its queries flawlessly. **C*** receives $(\text{PK}_i =< p_i, q_i, g_i, A_i, \hat{A}_i >)_{i\in\ell(k)}$ and R[**C**] as state information. Then **C*** builds the DHK$_\ell$ adversary $\mathcal{B}$ that runs the **sPA1**$_\ell$ adversary **C** internally and simulates the **sPA1**$_\ell$ experiment for it. $\mathcal{B}$ receives $(p_i, q_i, g_i, A_i)_{i\in\ell(k)}$ and it random coins from **C*** and parses its random coins as $(f_G^{-1}(\hat{A}_i))_{i\in[\ell(k)]}|R$[**C**] (prepared by **C*** where $\hat{A}_i$ is a random group element in G). Notice that since the group from which $\hat{A}_i$ is sampled is simulatable, it is arguable that $f_G^{-1}(\hat{A}_i)$ is indistinguishable from random bits and should have indistinguishable effect on the output of the extraction. Because $\mathcal{B}$ is a DHK$_\ell$ adversary, therefore there exists

an extractor for it $\mathcal{B}^*$. For each $i \in [\ell(k)]$, $\mathcal{B}$ sets $\text{PK}_i \leftarrow (p_i, q_i, g_i, A_i, \hat{A}_i)$. $\mathcal{B}$ then runs **C** on $(\text{PK}_i)_{i\in[\ell(k)]}$ and the random coins $R[\mathbf{C}]$ until **C** halts, answering to **C**'s queries as follows: upon receiving the query $C = (i, Y, W, U)$ from **C**, $\mathcal{B}$ submits (i, Y, W) to the DHK_ℓ extractor $\mathcal{B}^*$. The DHK_ℓ extractor $\mathcal{B}^*$ returns the value b. If $Y \not\equiv g_i^b \mod p_i$ or $W \not\equiv A_i^b \mod p_i$ then $\mathcal{B}$ returns $\perp$ as the answer to this query, otherwise $\mathcal{B}$ computes $M \leftarrow U \cdot (\hat{A}_i^{\,b})^{-1} \mod p_i$ and return the result to **C**. Notice that since $\mathcal{B}$ is a DHK_ℓ adversary, the extractor $\mathcal{B}^*$ should return the right answer to the queries $\mathcal{B}$ submits. Since $\mathbf{C}^*$ makes a mistake in answering **C**'s queries only when there is a mistake in $\mathcal{B}^*$'s answers to $\mathcal{B}$'s queries, we conclude that $\mathbf{C}^*$ also returns the right decryption to the queries submitted by **C** and is an extractor for it.

□

Theorem 3.2.10. *For any polynomial ℓ, The CS-Lite scheme is **sPA1**$_\ell$ secure under the DHK_ℓ assumption.*

Proof. Similar to the proof of Theorem 3.2.9.

□

3.2.3 Why sPA1$_\ell$ does not follow from sPA1 security

At first glance it seems that the **sPA1**$_\ell$ definition should follow naturally from **sPA1** by composing extractors. However, there are significant technical issues with the original definition that does not permit it to "naturally compose" with itself as one would expect. Specifically, issues emerge when considering how a ciphertext creator **C** interacts with decryption oracles—with respect to different keys—it might have at its disposal, and a corresponding extractor's ability to respond to these queries.

Consider an **sPA1**-secure primitive (g, e, d). We will modify it to a new encryption scheme, which has two pairs of keys from the original encryption scheme, and chooses which to use which to encrypt at random during the encryption process. Formally, (g', e', d') acts as follows $g'(k) = \big(PK = (pk_b)_{b\in\{0,1\}}, SK = (sk_b)_{b\in\{0,1\}}\big)$, where (pk_b, sk_b) are an output of the bth invocation of $g(k)$. For encryption, $e'(PK, m)$ chooses a random coin $z \in \{0, 1\}$ and outputs

$C = (z, e(\text{PK}_z, m))$; decryption is $d'(SK, C = (z, c))$ outputs $d(sk_z, c)$. One would expect that the resulting scheme is naturally **sPA1** secure, but it is not clear that it is. In particular, one would think that for any ciphertext creator for the modified scheme, one could just use two extractors for the original scheme (one for each public-key) to simulate an extractor for the creator. However, this argument does not work, and we are not aware of any other methods for proving the equivalence. One issue is that if a creator switches between making encryptions under PK_b and PK_{1-b}, then at each switch we must incorporate the extractor in to the original ciphertext creator in order to achieve a new extractor, and the result is a potential super-polynomial blow-up in the running time of the constructor after more than a constant number of such iterations. The extractors must be continuously incorporated, because definitionally they have no ability to extract encryptions when the ciphertext creator has access to a decryption oracle other than the one simulated by the extractor.

For example, consider a ciphertext creator C for the scheme (g', e', d') we just described. Let it switch between the public-key used to encrypt messages n times. Describe the creator $C = (C_0, C_1, .., C_n)$, where C_i denotes the execution after the ith switch.[1] Assume it makes it encrypts its even queries with PK_0 and its odd queries with PK_1. To make an extractor for C (without including the oracles in the definition, as we have done), we would first create C'_0 using the standard **sPA1** definition and the extractor for PK_0 that is guaranteed to exist for C_0, call it C^*_0, by embedding the extractor as a subroutine into C_0. The running time of C'_0 is clearly the additive combination of the running time of C^*_0 and C_0. One would then compose C'_0 with C_1 and use the **sPA1** definition to construct an extractor for $C_1 \circ C'_0$, called C^*_1, which only queries decryptions for PK_1. Intuitively, we would continue this inductively, but after a non-constant number of iterations the running time would be super-polynomial.

Finally, we note that common additional definitional traits, like the notion of a history of computation, do not port readily to these extractability definitions. In essence, one needs to consider the possibility that a history string encodes a turing machine, which is then run by an extractor acting as a Universal Turing machine. The semantic effect of such a notion in the definition is to

[1] We can assume the C_i outputs its state, which is then used as auxiliary information and passed as input to C_{i+1}.

swap the order of quantifiers relating to the extractor, further strengthening the definition.

3.2.4 A Note On PA1$^+$

Dent [29] also investigated an augmented notion of plaintext awareness in which he provides the ciphertext creator access to an oracle that produces random bits, PA1$^+$. The extractor receives the answers to any queries generated by the creator, but only at the time these queries are issued. The point of this oracle in the context of a plaintext awareness definition is to model the fact that the extractor might not receive all of the random coins used by the creator *at the beginning* of the experiment. Much in the spirit of "adaptive soundness" and "adaptive zero-knowledge", this oracle requires the extractor to work even when it receives the random coins at the same time as the ciphertext creator. Therefore, the extractor potentially needs to be able to extract some ciphertexts independent of future randomness. This modification has implications when the notion of plaintext awareness is computational—as in the case of Dent's work. However, in our case, we require statistical plaintext awareness, and as we argue below, allowing access to such an oracle does not affect the **sPA1**$_\ell$ security.

We claim that any encryption scheme that is **sPA1**$_\ell$ secure is also **sPA1**$_\ell^+$ secure.

Definition 3.2.11. *Define the* ***sPA1***$_\ell^+$ *experiment in a similar way to the* ***sPA1***$_\ell$ *experiment.* The only difference *between the two is that during the* ***sPA1***$_\ell^+$ *experiment, the ciphertext creator has access to a random oracle* $\mathcal{O}$ *that* takes no input*, but returns independent uniform random strings upon each access. Any time the creator access the oracle, the oracle's response is forwarded to both the creator and extractor.*

If an encryption scheme would be deemed ***sPA1***$_\ell$ *secure, when we replace the* ***sPA1***$_\ell$ *experiment in the definition with the modified* ***sPA1***$_\ell^+$ *experiment, then the encryption scheme is said to be* ***sPA1***$_\ell^+$ *secure.*

Lemma 3.2.12. *If an encryption scheme* Π *is* ***sPA1***$_\ell$ *secure, then it is* ***sPA1***$_\ell^+$ *secure.*

Proof. We prove for the case where the ciphertext creator accesses the oracle $\mathcal{O}$ only once and for the case where $\ell(k) = 1$. Generalizing the proof for the case for where the ciphertext creator

accesses $\mathcal{O}$ for polynomially many times is then straightforward. Also a very similar argument can be made for **sPA1**$^+_\ell$ case where ℓ is an arbitrary polynomial.

Let **Point** i denote the point right after the extractor answers the i^{th} query of the ciphertext creator in the **sPA1** experiment. Without loss of generality, assume the one time access (of $\mathcal{O}$ by the ciphertext creator) occurs at **Point** a. We refer to the string that $\mathcal{O}$ returns at **Point** a as r.

To prove the Lemma, we need to show that for any **sPA1**$^+$ adversary **C**, there exists an extractor **C*** that can decrypt the queries correctly. Notice that if we knew the string returned at **Point** a, then **C** would be an **sPA1** adversary for which there exists an extractor **C**$'$ that returns the right decryptions to its queries.

We build the extractor **C*** as follows. **C*** receives $(\text{PK}, R[\mathbf{C}])$ from the outside and samples a random string r' where $|r| = |r'|$. **C*** ,then, embeds r' in $R[\mathbf{C}]$ in a way that it gets accessed by **C** at **Point** a (but not before then). We call the resulting random tape $R'[\mathbf{C}]$. Then, **C*** runs **C** and **C**$'$ on the input $(\text{PK}, R'[\mathbf{C}])$ up to **Point** a. At this point **C*** queries $\mathcal{O}$ and learns r. At this point, the state information of **C**$'$ needs to be updated to be consistent with the value that $\mathcal{O}$ returns. Hence **C*** pauses the execution with **C** and repeats the mentioned execution (using the algorithm of **C** and **C**$'$, the public key, the adversary's random coins and r), this time with r as the string that will be returned at **Point** a up to after **Point** a (note that in this run, **C** asks exactly the same queries as in the previous run before **Point** a). Learning the right state information at this point, **C*** resumes running **C** and **C**$'$ (with the corrected state information) until **C** terminates . Since r and r' are sampled randomly, in both executions **C**$'$ should return the right decryption to the queries. Therefore **C*** can decrypt all of **C**'s queries and is an extractor for it. □

3.3 The Construction

Let $E = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ be any encryption scheme that is weakly simulatable and **sPA1**$_\ell$ secure. Then we construct the encryption scheme Π represented in Figure 3.6 that is a non-malleable $\mathsf{CCA1}$ encryption scheme. Let $\Sigma = (\mathsf{GenKey}, \mathsf{Sign}, \mathsf{Verify})$ be a strong one-time signature scheme such that on security parameter k the verification keys that are constructed have length k.

As a first step, we define an encryption scheme $E' = (\mathbf{Gen}', \mathbf{Enc}', \mathbf{Dec}')$ in which one encrypts the encryption of a message k times with k independently chosen public keys. More specifically:

— $\mathbf{Gen}'(1^k)$: For $i \in [0..k]$, run $(\mathrm{PK}_i, \mathrm{SK}_i) \leftarrow \mathbf{Gen}(1^k)$. Set the public and secret keys as $\mathrm{PK} \stackrel{def}{=} (\mathrm{PK}_0, \mathrm{PK}_1, \ldots, \mathrm{PK}_k)$ and $\mathrm{SK} \stackrel{def}{=} (\mathrm{SK}_0, \mathrm{SK}_1, \ldots, \mathrm{SK}_k)$

— $\mathbf{Enc}'_{\mathrm{PK}=\mathrm{PK}_0,..,\mathrm{PK}_k}(m)$: Output $[\mathbf{Enc}_{\mathrm{PK}_1}(\mathbf{Enc}_{\mathrm{PK}_0}(m; r_0); r_1), \ldots \mathbf{Enc}_{\mathrm{PK}_k}(\mathbf{Enc}_{pk_0}(m; r_0); r_k)]$ using independently chosen coins r_i.

— $\mathbf{Dec}'_{\mathrm{SK}=\mathrm{SK}_0,\ldots,\mathrm{SK}_k}([c_1, c_2, \ldots, c_k])$: Compute $(c'_i = \mathbf{Dec}_{\mathrm{SK}_i}(c_i))_{i\in[k]}$. If all c'_i are not equal, output $\perp$, else output $\mathbf{Dec}_{\mathrm{SK}_0}(c'_1)$.

We are now ready to present our main construction Π defined in Figure 3.6.

$\mathrm{NMGen}(1^k)$
1: $(\mathrm{PK}_0, \mathrm{SK}_0) \leftarrow \mathbf{Gen}(1^k)$; $(\mathrm{PK}_i^b, \mathrm{SK}_i^b) \leftarrow \mathbf{Gen}(1^k),\ \forall i \in [k]$ and $b \in \{0,1\}$
2: Output $\mathrm{NPK} = \{\mathrm{PK}, \mathrm{PK}_0\}$ and $\mathrm{NSK} = \{\mathrm{SK}, \mathrm{SK}_0\}$ where $\mathrm{PK} = \{(\mathrm{PK}_i^0, \mathrm{PK}_i^1)\}_{i\in[k]}$ and $\mathrm{SK} = \{(\mathrm{SK}_i^0, \mathrm{SK}_i^1)\}_{i\in[k]}$

$\mathrm{NMEnc}(\mathrm{NPK} = (\mathrm{PK}, \mathrm{PK}_0), m)$
1: $(\mathsf{SigSK}, \mathsf{SigVK}) \leftarrow \mathsf{GenKey}(1^k)$
2: $c \leftarrow \mathbf{Enc}'_{\mathrm{PK}_0, \mathrm{PK}_1^{\mathsf{SigVK}_1}, \ldots, \mathrm{PK}_k^{\mathsf{SigVK}_k}}(m)$
3: $\sigma \leftarrow \mathsf{Sign}_{\mathsf{SigSK}}(c)$
4: Output $(c, \mathsf{SigVK}, \sigma)$

$\mathrm{NMDec}(\mathrm{NSK} = (\mathrm{SK}, \mathrm{SK}_0), C = (c, \mathsf{SigVK}, \sigma))$
1: **if** $\mathsf{Verify}_{\mathsf{SigVK}}(\sigma, c) = 0$ **then** Output $\perp$
2: Output $\mathbf{Dec}'_{\mathrm{SK}_0, \mathrm{SK}_1^{\mathsf{SigVK}_1}, \ldots, \mathrm{SK}_k^{\mathsf{SigVK}_k}}(c_1)$

Figure 3.6: THE NON-MALLEABLE CCA1 ENCRYPTION SCHEME Π

Lemma 3.3.1. *If $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ is weakly simulatable, then $\boldsymbol{E}' = (\boldsymbol{Gen}', \boldsymbol{Enc}', \boldsymbol{Dec}')$ is weakly simulatable as well.*

Proof. We argue via a standard hybrid argument. We assume that there exists an adversary that can distinguish between real ciphertexts and pseudo-ciphertexts for the encryption scheme $\mathbf{E}'$ with

advantage ϵ. Then we build an adversary $\mathcal{A}$ that has advantage of at least $\epsilon/(k+1)$ in distinguishing real ciphertexts and pseudo-ciphertexts for the encryption scheme $\mathbf{E}$.

We define a series of hybrids $\{\mathcal{H}_i\}_{i\in[k+1]}$. In $\mathcal{H}_i$ for $i \in [k]$, the pair of random coins and ciphertexts (r, c) in the weakly simulatability experiment for message m is computed as follows:

$$c = (f(\mathrm{PK}_1, r_1), f(\mathrm{PK}_2, r_2), \ldots, f(\mathrm{PK}_{i-1}, r_{i-1}), \mathbf{Enc}_{\mathrm{PK}_i}(\mathbf{Enc}_{\mathrm{PK}_0}(m; r_0); r_i), \ldots, \mathbf{Enc}_{\mathrm{PK}_k}(\mathbf{Enc}_{\mathrm{PK}_0}(m; r_0); r_k))$$

and

$$r = (r_1, r_2, \ldots, r_{i-1}, f^{-1}(\mathrm{PK}_i, c_i), \ldots, f^{-1}(\mathrm{PK}_k, c_k))$$

where $\{r_i\}_{i\in[0..k]}$ are independent random coins. Notice that c in $\mathcal{H}_1$ is a real ciphertext and in $\mathcal{H}_k$ is a pseudo ciphertext for the encryption scheme $\mathbf{E}'$. By hybrid lemma, if there exists the adversary $\mathcal{B}$ that can distinguish between $\mathcal{H}_1$ and $\mathcal{H}_k$, then there exists $i \in [k]$ such that $\mathcal{B}$ distinguishes $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ with advantage at least $\epsilon/(k+1)$. Then we build the adversary $\mathcal{A}$ that distinguishes real and pseudo ciphertexts of encryption scheme $\mathbf{E}$ with advantage at least $\epsilon/(k+1)$.

The reduction works as follows: $\mathcal{A}$ receives a public key, which we call PK_i from the outside. $\mathcal{A}$ generates another k public keys $(\mathrm{PK}_1, \mathrm{PK}_2, \ldots, \mathrm{PK}_{i-1}, \mathrm{PK}_{i+1}, \ldots, \mathrm{PK}_k)$ and PK_0 using the key generation algorithm and sends $(\mathrm{PK}_0, \mathrm{PK}_1, \ldots, \mathrm{PK}_k)$ to $\mathcal{B}$. Then $\mathcal{B}$ outputs a message m'. $\mathcal{A}$ computes $m = \mathbf{Enc}_{\mathrm{PK}_0}(m'; r_0)$ where r_0 is random coins and outputs it to the outside. Then $\mathcal{A}$ receives from the outside a pair of strings which we call (r_i, c_i). $\mathcal{A}$ then prepares the pair (r, c) for $\mathcal{B}$ as directed in $\mathcal{H}_i$ and replaces the i^{th} element with (r_i, c_i). $\mathcal{A}$ then forwards (c, r) to $\mathcal{B}$ and receives a guess bit b' and forwards $1 - b'$ to the outside as its guess for the bit b (we should negate the bit b' because of the way we defined the hybrids).

□

Theorem 3.3.2. *If* $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *is an encryption scheme that is weakly simulatable and also* $\boldsymbol{sPA1}_{\ell(k)=2k+1}$ *secure where* k *is the security parameter, then the encryption scheme* Π *as described in Figure 3.6 is a non-malleable* CCA1 *encryption scheme.*

Proof. Recall that Lemma 3.2.12 shows that if **E** is $\mathbf{sPA1}_\ell$ secure, then it is also $\mathbf{sPA1}_\ell^+$ secure. In what follows, the $\mathbf{sPA1}_\ell^+$ ciphertext creator adversaries always have access to an oracle $\mathcal{O}$ that produces random strings upon access.

To prove that Π is a non-malleable CCA1 encryption scheme, we need to show that for any ppt adversary $\mathcal{A}$ and ppt distinguisher $\mathcal{D}$ and for all polynomials $p(k)$,

$$\{\mathsf{NME}_0\left(\Pi, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\}_{k\in\mathbb{N}} \approx_c \{\mathsf{NME}_1\left(\Pi, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\}_{k\in\mathbb{N}}$$

We show this by a hybrid argument. Consider the following experiments:

Experiment $\mathsf{NME}_b{}^{(1)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ modifies NME_b in two ways. First, instead of selecting vkSig^* when the challenge ciphertext is encrypted, choose this value as the first step of the experiment. Second, when processing decryption queries during the experiment, replace Verify with Verify^* as follows:

Verify* Let vkSig^* be the verification key in the challenge ciphertext $(c^*, \sigma^*, \text{vkSig}^*)$. Upon receiving a decryption query on $(c, \sigma, \text{vkSig})$, output $\perp$ if either $\text{vkSig} = \text{vkSig}^*$ or $\text{Verify}_{\text{vkSig}}(c, \sigma) = 0$.

Claim 5. *For* $b \in \{0, 1\}$, $\{\mathsf{NME}_b\left(\Pi, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\}_{k\in\mathbb{N}} \approx_c \{\mathsf{NME}_b{}^{(1)}\left(\Pi, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\}_{k\in\mathbb{N}}$

Proof. Follows using standard techniques from the security of the signature scheme. □

Experiment $\mathsf{NME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ modifies $\mathsf{NME}_b{}^{(1)}$ to use an extractor to decrypt the inner layer ciphertext for the decryptions in the final parallel decryption query. Specifically, in $\mathsf{NME}_b{}^{(2)}$ the calls are submitted to NMDec^* as described below. This is unlike $\mathsf{NME}_b{}^{(1)}$ where the final ciphertexts $d_1, ..., d_{p(k)}$ are presented by A_2 for parallel decryption via calls to NMDec, :

NMDec$^*(d_i = \vec{C}, \sigma, \mathbf{vkSig})$ If $\text{Verify}^*_{\text{vkSig}}(\vec{C}, \sigma) = 0$ output $\perp$. For $i \in [k]$, do $C'_i \leftarrow \mathbf{Dec}_{\text{SK}_i^{\text{vkSig}_i}}(C_i)$ If $\exists_j, C'_1 \neq C'_j$, output $\perp$. Use the extractor $C^*_{\mathcal{A}}$ (defined in Lemma 3.3.3) to extract C'_i, where i is the smallest value s.t. $\text{vkSig}_i \neq \text{vkSig}^*_i$, where vkSig^* is the verification key of the

challenge ciphertext. Return the extracted plaintext.

Lemma 3.3.3. *For* $b \in \{0,1\}$, $\{\mathsf{NME}_b{}^{(1)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))\}_{k\in\mathbb{N}} \approx_c \{\mathsf{NME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))\}_{k\in\mathbb{N}}$

This lemma might, on first glance, seem to follow immediately because the whole purpose of the extractor is that it be able to simulate a decryption oracle. However, since the adversary has (i) seen the challenge ciphertext, (ii) it is not aware of the randomness used to produce this ciphertext, and (iii) created final parallel decryption queries potentially based on the challenge ciphertext, there is no a priori reason to believe the **sPA1** extractor will "decrypt" properly. However, we are only extracting on the inner layers of ciphertexts, and the inner layer of the challenge ciphertext has been hidden by the encryptions on the outer layer. Further, the outer layer is weakly simulatable, so we can argue that these new ciphertexts issued for parallel decryption, described in point (iii) above, are not dependent on the randomness of the inner-layer of the challenge ciphertext. Therefore, the extractor will function correctly.

Proof. The experiments differ only if the extractor returns a result that is different from the of the decryption oracle. We define **badExtract** to capture this event, and show that it occurs with negligible probability. Assume (for contradiction) that there exists an adversary $\mathcal{A}$ which induces the event **badExtract** to occur with non-negligible probability. We show that **E** is not a weakly simulatable encryption scheme.

Note that the public- and secret-keys for Π are composed of $2k+1$ keys that are generated using the key generation algorithm for encryption scheme **E**. To encrypt a message m, we first generate a pair of signing keys, $(\text{vkSig}, \text{skSig})$, and then encrypt m with a fixed public key, PK_0, in the set of $2k+1$ public keys (we refer to this ciphertext as the inner layer). Then, we select a subset of size k out of the $2k$ remaining keys determined by the bits of vkSig, and encrypt the inner layer using fresh random coins for k times under those k keys (we refer to these k ciphertexts as the outer layer). We refer to the key used to encrypt the inner layer as the inner key, and the remaining $2k$ keys as the outer keys.

The technical difficulty in showing that **badExtract** does not occur in the $\mathsf{NME}_b^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment is that by providing the challenge ciphertext, we actually provide the adversary with ciphertexts that are encrypted using k keys out of the $2k$ outer keys. We must argue that even in this case, there should be a way to extract the plaintext of the queries submitted by the adversary on the spots in the outer layer that are encrypted under a new key from the k keys used in the outer layer of the challenge ciphertext.

To do so, we first construct an $\mathbf{sPA1}_\ell^+$ ciphertext creator $\mathbf{C}_\mathcal{A}$ using the adversary $\mathcal{A}$. Since the encryption scheme $\mathbf{E}$ is $\mathbf{sPA1}_\ell^+$ secure, there exists an extractor for $\mathbf{C}_\mathcal{A}$ which we call $\mathbf{C}_\mathcal{A}^*$. Then we define a series of hybrids using $\mathbf{C}_\mathcal{A}$ and $\mathbf{C}_\mathcal{A}^*$, that are indistinguishable assuming $\mathbf{E}$ is weakly simulatable. The last hybrid in that series perfectly simulates the $\mathsf{NME}_b^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment for $\mathcal{A}$ up to the point when $\mathcal{A}$ returns the vector of the ciphertexts after receiving the challenge ciphertext. Based on the indistinguishability of the hybrids, we will argue that there exists an extractor that can decrypt the adversary's queries on the first spot i where $\text{vkSig}_i \neq \text{vkSig}_i^*$ with overwhelming probability. Notice that the extractor cannot be used to decrypt the outer layer on the spots where $\text{vkSig}_i = \text{vkSig}_i^*$, otherwise it could be argued that the encryption scheme $\mathbf{E}$ is indeed **PA2** secure (**PA2** security is defined in [8]) and hence CCA2 secure.

First we construct an $\mathbf{sPA1}_\ell^+$ ciphertext creator $\mathbf{C}_\mathcal{A}$ from $\mathcal{A}$ where $\ell = 2k+1$. $\mathbf{C}_\mathcal{A}$ interacts with the $\mathbf{sPA1}_\ell^+$ experiment in "the outside" as follows:

- $\mathbf{C}_\mathcal{A}$ receives $2k+1$ public keys $\left(\{\text{PK}'_i\}_{i\in[0\ldots2k]}\right)$ from the $\mathbf{sPA1}_\ell^+$ experiment. It generates a pair of signing keys $(\text{vkSig}^*, \text{skSig}^*) \leftarrow \mathsf{GenKey}(1^k)$ internally and sets $\text{PK} = \left(\{\text{PK}_i^\alpha\}_{i\in[0\ldots2k],\alpha\in\{0,1\}}\right)$ as described (intuitively, $\mathbf{C}_\mathcal{A}$ arranges PK such that it can potentially sign a vector of ciphertexts that are supposed to be encrypted under the last k keys in $\vec{\text{PK}}'$, $\left(\text{PK}'_{k+1}, \ldots, \text{PK}'_{k+k}\right)$, to generate a valid ciphertext in the Π scheme):

$$\text{for } i \in [0\ldots k] \;\&\; \alpha \in \{0,1\}, \text{PK}_i^\alpha = \begin{cases} \text{PK}'_0 & \text{if } i = 0 \\ \text{PK}'_i & \text{else if } \text{vkSig}_i^* \neq \alpha \\ \text{PK}'_{i+k} & \text{otherwise} \end{cases}$$

$\mathbf{C}_{\mathcal{A}}$ runs $\mathcal{A}_1$ on the input of PK. Note that this rearrangement of keys is crucial to make the view of the adversary $\mathcal{A}$ on the arrangement of the keys identical to its view in a real $\mathsf{NME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment. To see this, consider the following example. The adversary $\mathcal{A}$ might abort whenever the keys used in the outer layer of the challenge ciphertext are the last k keys in $\vec{\text{PK}}$. Such a coincidence occurs in the simulated $\mathsf{NME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment with probability 1 if $\mathbf{C}_{\mathcal{A}}$ sets $\vec{\text{PK}}$ to be the same as $\vec{\text{PK}}'$, while this coincidence occurs in a real $\mathsf{NME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment with negligible probability due to the security of the signature scheme.

- Whenever $\mathbf{C}_{\mathcal{A}}$ receives a query $\big(\{y_i\}_{i\in[k]}, \sigma, \text{vkSig}\big)$ from $\mathcal{A}_1$, it first checks if the signature is valid. If not, it returns $\perp$ as the answer to this query. Next, it checks whether $\text{vkSig} = \text{vkSig}^*$. If so, it aborts. Otherwise, $\mathbf{C}_{\mathcal{A}}$ submits y_i's one by one to the extractor. If all of the queries do not get decrypted to the same value, $\mathbf{C}_{\mathcal{A}}$ returns $\perp$ to $\mathcal{A}_1$ as the answer to that query. But if all of the queries get decrypted to the same value y_0, $\mathbf{C}_{\mathcal{A}}$ then submits y_0 (which is supposed to be an encryption under PK_0^0) to the extractor and returns the result to $\mathcal{A}_1$. Eventually $\mathcal{A}_1$ returns (m_0, m_1, St) and halts. $\mathbf{C}_{\mathcal{A}}$ outputs (m_0, m_1).

- $\mathbf{C}_{\mathcal{A}}$ accesses its oracle $\mathcal{O}$ and generates k blocks of random bits of length l, giving the vector $\vec{x} = (x_1, \ldots, x_k)$. Let $\vec{y} = \big(f(\text{PK}'_{k+1}, x_1), \ldots, f(\text{PK}'_{k+k}, x_k)\big)$. $\mathbf{C}_{\mathcal{A}}$ then computes $\sigma^* = \mathsf{Sign}(\vec{y}, \text{skSig}^*)$, and runs $\mathcal{A}_2$ on the input $y^* = (\vec{y}, \sigma^*, \text{vkSig}^*)$ and St.

- $\mathcal{A}_2$ returns a vector of ciphertexts $\vec{Y}$ and the state information S and halts. For all $j \in [|\vec{Y}|]$, $\mathbf{C}_{\mathcal{A}}$ does the following: on the query $Y_j = (\{y_i\}_{i\in[k]}, \sigma, \text{vkSig})$, it first checks if the signature is valid. If not, it moves to the next query. Otherwise, it checks whether $\text{vkSig} = \text{vkSig}^*$. If so, it aborts. Otherwise, $\mathbf{C}_{\mathcal{A}}$ finds the first index i where $\text{vkSig}_i \neq \text{vkSig}_i^*$, and submits y_i to its extractor to be "decrypted" under $\text{PK}_i^{\text{vkSig}_i}$. $\mathbf{C}_{\mathcal{A}}$ then submits the answer from the extractor (which is supposed to be an encryption under PK_0^0) again to the extractor to be "decrypted" under PK_0^0. Denote the result m'_j.

  $\mathbf{C}_{\mathcal{A}}$ returns $\{Y_j, m'_j\}_{j\in[|\vec{Y}|]}$ and the state information S, it halts.

Since $\mathbf{C}_{\mathcal{A}}$ is a $\mathbf{sPA1}^{+}_{\ell}$ ciphertext creator adversary, the $\mathbf{sPA1}^{+}_{\ell}$ security of $\mathbf{E}$ implies there exists an extractor $\mathbf{C}^{*}_{\mathcal{A}}$ whose answers to the decryption queries submitted by $\mathbf{C}_{\mathcal{A}}$ are indistinguishable from their true decryptions. We call the above interaction **Game 1**. Let $\Pr[W_i]$ be the probability of the adversary $\mathbf{C}_{\mathcal{A}}$ inducing the event **badExtract** in the **Game** i. The $\mathbf{sPA1}^{+}_{\ell}$ security implies that $\Pr[W_1]$ is bounded by $\mathbf{Adv}^{\mathbf{sPA1}^{+}_{\ell}}(\mathbf{E}, \mathbf{C}_{\mathcal{A}}, \mathbf{C}^{*}_{\mathcal{A}}, k)$ which is negligible in k. Hence:

$$\Pr[W_1] \leq \mathbf{Adv}^{\mathbf{sPA1}^{+}_{\ell}}(\mathbf{E}, \mathbf{C}_{\mathcal{A}}, \mathbf{C}^{*}_{\mathcal{A}}, k) \tag{3.4}$$

We will define another game, **Game 2**, which is identical to **Game 1** with the difference that instead of a fake ciphertext, $\mathcal{A}_2$ is fed with a real ciphertext as the challenge ciphertext. The aborting probability of $\mathcal{A}_2$ in **Game 1** and **Game 2** is negligibly close otherwise it can be argued that $\mathbf{E}$ is not weakly simulatable. In what follows, we only deal with the probability of inducing the event **badExtract**. Also notice that **Game 2** simulates $\mathsf{NME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ for the adversary $\mathcal{A}$ up to the point when the adversary $\mathcal{A}$ returns a vector of ciphertexts after seeing the challenge ciphertext. That is because $\mathbf{C}_{\mathcal{A}}$ only needs the vector of the ciphertext generated by $\mathcal{A}$ after revealing the challenge ciphertext to induce the event **badExtract** to occur. After receiving such a vector of ciphertexts, $\mathbf{C}_{\mathcal{A}}$ does not need to complete the simulation of the $\mathsf{NME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment for $\mathcal{A}$.

In **Game 2** we modify the oracle $\mathcal{O}$ as follows: when $\mathbf{C}_{\mathcal{A}}$ accesses the oracle $\mathcal{O}$ for the i^{th} time, instead of $r \in \{0,1\}^l$, $\mathcal{O}$ returns $f^{-1}(\mathrm{PK}'_{k+i}, \mathbf{Enc}_{\mathrm{PK}'_{k+i}}(m_d))$ where m_d is picked randomly out of the two messages returned by $\mathcal{A}$. During **Game 2**, the random bit d is fixed. We argue that such a change does not affect the advantage of $\mathbf{C}_{\mathcal{A}}$ in inducing the event **badExtract** as otherwise $\mathbf{E}$ is not weakly simulatable countering our assumption. Using $\mathbf{C}_{\mathcal{A}}$ and $\mathbf{C}^{*}_{\mathcal{A}}$, we build the attacker $\mathcal{B}$ that distinguishes $(r, f(., r))$ and $(f^{-1}(., c = \mathbf{Enc}(., .)), c)$ as follows:

1. The challenger samples k pairs of random keys $(\mathrm{PK}_i, \mathrm{SK}_i) \leftarrow \mathbf{Gen}(1^k)$ for $1 \leq i \leq k$, and a random bit b.

2. The attacker $\mathcal{B}$ receives $\{\text{PK}_i\}_{i\in[k]}$. $\mathcal{B}$ then samples $k+1$ other random keys $(\text{PK}'_i, \text{SK}'_i) \leftarrow \textbf{Gen}(1^k)$ for $0 \leq i \leq k$. Let $\vec{pk''} = (\text{PK}'_0, \text{PK}'_1, \ldots, \text{PK}'_k, \text{PK}_1, \text{PK}_2, \ldots, \text{PK}_k)$. $\mathcal{B}$ samples random coins for $\mathbf{C}_\mathcal{A}$ and $\mathbf{C}^*_\mathcal{A}$ and sets $st \leftarrow (\vec{\text{PK}''}, R[\mathbf{C}_\mathcal{A}])$. $\mathcal{B}$ runs $\mathbf{C}_\mathcal{A}$ on the input $\vec{pk''}$, and $\mathbf{C}^*_\mathcal{A}$ on the input st. Eventually $\mathbf{C}_\mathcal{A}$ outputs (m_0, m_1). $\mathcal{B}$ randomly chooses $d \in \{0,1\}$ and outputs $c'_d = \textbf{Enc}_{\text{PK}'_0}(m_d)$. The challenger samples $r_i \in \{0,1\}^l$ for $1 \leq i \leq k$ and returns $\{(r_i, f(\text{PK}_i, r_i))\}_{i\in[k]}$ if $b = 0$, and $\{(f^{-1}(\text{PK}_i, c_i = \textbf{Enc}_{\text{PK}_i}(c'_d)), c_i)\}_{i\in[k]}$ if $b = 1$. Call the resulting vector (given by the challenger) $\vec{y}$. $\mathcal{B}$ then forwards $\{f^{-1}(\text{PK}_i, y_i)\}_{i\in[k]}$ to $\mathbf{C}_\mathcal{A}$ and $\mathbf{C}^*_\mathcal{A}$ when $\mathbf{C}_\mathcal{A}$ queries $\mathcal{O}$ for the i^{th} time. After $\mathbf{C}_\mathcal{A}$ halts, the attacker $\mathcal{B}$ checks if all the queries made by $\mathbf{C}_\mathcal{A}$ to the extractor after outputting m_0 and m_1 were answered correctly. This is done by using the extractor using $\vec{\text{SK}'}$ (notice that $\mathbf{C}_\mathcal{A}$ was made in a way that after returning m_0 and m_1, it always only asks the extractor on the ciphertexts encrypted under $\vec{pk'}$ which are the first $k+1$ keys in $\vec{pk}$). If so it outputs $b' = 0$ otherwise $b' = 1$.

When $b = 0$, **Game 1** is being simulated, and when $b = 1$, **Game 2** is being simulated. Therefore:

$$\begin{aligned}\Pr[b' = b] &= \Pr[b = 0] \cdot \Pr[b' = b | b = 0] + \Pr[b = 1] \cdot \Pr[b' = b | b = 1] \\ &= \frac{1}{2} \cdot (1 - \Pr[W_1]) + \frac{1}{2} \cdot \Pr[W_2]\end{aligned}$$

On the other hand, by Lemma 3.3.1, the advantage of the attacker $\mathcal{B}$ in guessing the bit b is negligible in k, and hence there exists a negligible function $\epsilon_1(.)$ such that $\Pr[b' = b] \leq \frac{1}{2} + \epsilon_1(k)$. Therefore:

$$\Pr[b' = b] = \frac{1}{2} \cdot (1 - \Pr[W_1]) + \frac{1}{2} \cdot \Pr[W_2] \leq \frac{1}{2} + \epsilon_1(k)$$

$$\implies \Pr[W_2] \leq 2 \cdot \epsilon_1(k) + \Pr[W_1] \tag{3.5}$$

$$\implies \Pr[W_2] \leq 2 \cdot \epsilon_1(k) + \mathbf{Adv}^{\mathbf{sPA1}^{+}_{\ell}}(\mathbf{E}, \mathbf{C}_{\mathcal{A}}, \mathbf{C}^{*}_{\mathcal{A}}, k) \tag{3.6}$$

Inequality (3.6) follows from Inequalities (3.4) and (3.5). Therefore $\Pr[W_2]$ is negligible. Since $\Pr[W_2]$ is the probability that the event **badExtract** occurs, we conclude that there is a negligible chance that **badExtract** occurs. Hence:

$$\{\mathsf{NME}_b^{(1)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))\}_{k \in \mathbb{N}} \approx_c \{\mathsf{NME}_b^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))\}_{k \in \mathbb{N}}$$

□

Lemma 3.3.4. *For every ppt adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a ppt adversary $\mathcal{B}$ such that for $b \in \{0, 1\}$,*

$$\{\mathsf{NME}_b^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))\}_{k \in \mathbb{N}} \equiv \{\mathsf{CPA}_b(\boldsymbol{E}, \mathcal{B}, k)\}_{k \in \mathbb{N}}$$

Proof. In the proof of Lemma 3.3.3, we showed how to construct the ciphertext creator $\mathbf{C}_{\mathcal{A}}$ that runs $\mathcal{A}$ internally and proved that there exists an extractor $\mathbf{C}^{*}_{\mathcal{A}}$ that can decrypt the queries submitted by $\mathbf{C}_{\mathcal{A}}$ with overwhelming probability.

We build the CPA adversary $\mathcal{B}$ that interacts with the CPA experiment. Having the algorithms for $\mathcal{A}$, $\mathbf{C}_{\mathcal{A}}$ and $\mathbf{C}^{*}_{\mathcal{A}}$, the CPA adversary $\mathcal{B}$ acts as follows: $\mathcal{B}$ receives the public key PK' from the CPA experiment, and generates $2k$ keys as $(\mathrm{PK}''_i, \mathrm{SK}''_i) \leftarrow \mathbf{Gen}(1^k)$ for $i \in [2k]$. Let $\vec{\mathrm{PK}} = \left(\mathrm{PK}', \vec{\mathrm{PK}}''\right)$. $\mathcal{B}$ runs $\mathbf{C}_{\mathcal{A}}$ (that simulates $\mathcal{A}$ internally) on $\vec{\mathrm{PK}}$ and its random coins. Whenever $\mathbf{C}_{\mathcal{A}}$ asks a query, $\mathcal{B}$ runs $\mathbf{C}^{*}_{\mathcal{A}}$ to answer them ($\mathbf{C}^{*}_{\mathcal{A}}$ gets to know the random coins of $\mathbf{C}_{\mathcal{A}}$ and all of its input as described in the proof of Lemma 3.3.3). Eventually $\mathbf{C}_{\mathcal{A}}$ outputs (m_0, m_1). $\mathcal{B}$ outputs m_0 and m_1 to the CPA experiment, and receives a ciphertext y. Remember that $\mathbf{C}_{\mathcal{A}}$ now accesses the oracle $\mathcal{O}$ k times. Using fresh random coins for each encryption, $\mathcal{B}$ computes $C_i = \mathbf{Enc}_{\mathrm{PK}_{k+i}}(y)$ and sends

$f^{-1}(\text{PK}_{k+i}, C_i)$ to $\mathbf{C}_{\mathcal{A}}$ (and $\mathbf{C}^*_{\mathcal{A}}$) on the i^{th} access to $\mathcal{O}$. Eventually $\mathbf{C}_{\mathcal{A}}$ returns $\{Y_i, m'_i\}_{i\in[|\vec{Y}|]}$ and the state information S and halts. The only step left in determining the decryption of Y_i is to decrypt all the ciphertexts in the outer layer, and check that they all decrypt to the same value. $\mathcal{B}$ has the $2k$ secret keys for the outer layer, hence it can do the mentioned check. If the outer layer ciphertexts of Y_i do not decrypt to the same value, the decryption of Y_i is $\perp$, otherwise the decryption of Y_i is m'_i. After $\mathcal{B}$ decrypts all the Y_i, it submits the results along with the state information S to the distinguisher $\mathcal{D}$ and forwards $\mathcal{D}$'s output to the CPA experiment.

□

□

3.4 More Than Non-Malleable CCA1 Encryption Scheme

In the previous section, we showed how to build a non-malleable CCA1 encryption scheme from any encryption scheme that is weakly simulatable and **sPA1**$_\ell$ secure. Define a parallel query as a query consisting of unbounded number of ciphertexts, none of which will be decrypted until all the ciphertexts in the query are submitted. In the NM-CCA1 game, the adversary is allowed to ask an unbounded number of queries before seeing the challenge ciphertext, and one parallel query afterwards. This compares with CCA2 secure encryption schemes, which are secure even if the adversary asks an unbounded number of queries before and after seeing the challenge ciphertext. The NM-CCA1 constructions seem to be much weaker primitives. However, between the extremes of the NM-CCA1 security and the CCA2 security, a range of security notions can be defined that distinguish themselves based on how many queries the adversary may ask after revealing the challenge ciphertext without sacrificing indistinguishability of ciphertexts.

Define cNM-CCA1 security identically to NM-CCA1 security except that the adversary can make $c \geq 1$ parallel queries after seeing the challenge ciphertext. We show how to extend our result to construct an encryption scheme that is cNM-CCA1 secure where c is a constant. The high level idea for constructing a cNM-CCA1 scheme is to add another c layers of encryption on top of the ciphertext from the previous section. Intuitively, with the first parallel query, the adversary can only

ask queries that can help it to maul the first layer of encryption from the outside in the future. In other words, with the first parallel query, the adversary can gain no information about all the inner ciphertexts. Hence, to penetrate the innermost layer, the adversary has to ask at least c parallel queries. Notice also that this type of construction can only allow a *constant* c since each layer of encryption increases the key size, the encryption and decryption time and the ciphertext size by a polynomial factor. Hence only when c is a constant, these factors would still be polynomial in the security parameter.

We start with providing the definition for cNM-CCA1 security and then we show how an cNM-CCA1 secure encryption can be constructed for the constant c.

Definition 3.4.1 (cNM-CCA1)**.** *We say that* $\Pi^{(c)} = (NMGen^c, NMEnc^c, NMDec^c)$ *is cNM-CCA1 or* (c)NME *secure if for all ppt adversaries and distinguishers* $\mathcal{A}$ *and* $\mathcal{D}$ *respectively if* $\{\text{(c)NME}_0(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\}$ $\{\text{(c)NME}_1(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\}_k$ *where experiment* (c)NME *is defined in Figure 3.7.*

$\text{(c)NME}_b(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$

1: $(\text{CNPK}, \text{CNSK}) \leftarrow \text{NMGen}^{(c)}(1^k)$
2: $(m_0, m_1, S_1) \leftarrow A_0^{\text{NMDec}^{(c)}(\text{CNSK},.)}(\text{CNPK})$ s.t. $|m_0| = |m_1|$
3: $y \leftarrow \text{NMEnc}^{(c)}(\text{CNPK}, m_b)$
4: $\vec{d_1} \leftarrow \perp$
5: *for* $i = 1$ *to* c
6: $\quad (\vec{c}, S_{i+1}) \leftarrow \mathcal{A}_i(y, S_i, \vec{d_i})$ where $|\vec{c}| = p(k)$
7: $\quad d_{i+1,j} \leftarrow \text{NMDec}^{(c)}(\text{CNSK}, c_j)$ if $c_j \neq y$ o/w $d_{i+1,j} \leftarrow \perp; \forall j \in [|\vec{c}|]$
8: Output $\mathcal{D}(\vec{d}_{c+1}, S_{c+1})$

Figure 3.7: THE (c)NME EXPERIMENT

3.4.1 The Construction

Figure 3.8 shows a cNM-CCA1 construction that is based on the NM-CCA1 construction presented in Figure 3.6.

Theorem 3.4.2. *Given that* $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *is weakly simulatable and* $\boldsymbol{sPA1}_\ell$ *secure, the construction* $\Pi^{(c)} = (NMGen^{(c)}, NMEnc^{(c)}, NMDec^{(c)})$ *shown in Figure 3.8 is cNM-CCA1 secure.*

We define cNM-CCA1 recursively. As for the base case, we define $(\text{NMGen}^{(0)}, \text{NMEnc}^{(0)}, \text{NMDec}^{(0)})$ equivalently to $(\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

$\text{NMGen}^{(c)}(1^k)$
1: $(\text{NPK}^{(c-1)}, \text{NSK}^{(c-1)}) \leftarrow \text{NMGen}^{(c-1)}(1^k)$
2: $(\text{PK}_i^b, \text{SK}_i^b) \leftarrow \textbf{Gen}(1^k),\ \forall i \in [k]$ and $b \in \{0,1\}$ where the length of the message the PK_i^b key can encrypt is the same as the size of a ciphertext in $\Pi^{(c-1)}$ without the signature and the verification key
3: Output $\text{NPK}^{(c)} = \{\text{NPK}^{(c-1)}, \vec{\text{PK}}\}$ and $\text{NSK}^{(c)} = \{\text{NSK}^{(c-1)}, \vec{\text{SK}}\}$

$\text{NMEnc}^{(c)}(\text{NPK}^{(c)}, m)$
1: $(\mathsf{SigSK}, \mathsf{SigVK}) \leftarrow \mathsf{GenKey}(1^k)$
2: $c \leftarrow \text{NMEnc}^{(c)}(\text{NPK}^{(c)}, m, \mathsf{SigVK})$
3: $\sigma \leftarrow \mathsf{Sign}_{\mathsf{SigSK}}(c)$
4: Output $C = (c, \mathsf{SigVK}, \sigma)$

$\text{NMEnc}^{(c)}(\text{NPK}^{(c)}, m, \mathsf{SigVK})$
1: Parse $\text{NPK}^{(c)}$ into $(\text{NPK}^{(c-1)}, \vec{pk} = \{\text{PK}_1^b, \ldots, \text{PK}_k^b\}_{b\in\{0,1\}})$
2: $c_0'' \leftarrow \text{NMEnc}^{(c-1)}(\text{NPK}^{(c-1)}, m, \mathsf{SigVK})$
3: $c_i' \leftarrow \textbf{Enc}_{\text{PK}_i^{\mathsf{SigVK}_i}}(c_0''); \forall i \in [k]$
4: Output $c = \vec{c'}$

$\text{NMDec}^{(c)}(\text{NSK}^{(c)}, C = (c, \mathsf{SigVK}, \sigma))$
1: **if** $\mathsf{Verify}_{\mathsf{SigVK}}(\sigma, \vec{c}) = 0$ **then** Output $\perp$
2: Output $\text{NMDec}^{(c)}(\text{NSK}^{(c)}, c, \mathsf{SigVK})$

$\text{NMDec}^{(c)}(\text{NSK}^{(c)}, c, \mathsf{SigVK})$
1: Parse $\text{NSK}^{(c)}$ into $(\text{NSK}^{(c-1)}, \vec{sk} = \{\text{SK}_1^b, \ldots, \text{SK}_k^b\}_{b\in\{0,1\}})$
2: $c' \leftarrow \textbf{Dec}_{\text{SK}_1^{\mathsf{SigVK}_1}}(c_1)$
3: **if** $\exists i \in [k]$ s.t. $c' \neq \textbf{Dec}_{\text{SK}_i^{\mathsf{SigVK}_i}}(c_i)$ **then** Output $\perp$
4: Output $\text{NMDec}^{(c-1)}(\text{NSK}^{(c-1)}, c', \mathsf{SigVK})$

Figure 3.8: THE CNM-CCA1 ENCRYPTION SCHEME $\Pi^{(c)}$

Note that cNM-CCA1 security definition is equivalent to NM-CCA1 security definition when $c = 1$. Also the encryption scheme $\Pi^{(c)}$ (presented in Figure 3.8) is equivalent to the encryption scheme Π (presented in Figure 3.6) when $c = 1$. We will prove the cNM-CCA1 security for $\Pi^{(c)}$ by induction. The base case is when $c = 1$ which we already proved. Then we prove the cNM-CCA1

security for the cases $c \geq 2$. As the inductive hypothesis, we assume the encryption scheme $\Pi^{(c-1)}$ is $(c-1)$NM-CCA1 secure in the proof of cNM-CCA1 security.

Proof. In Lemma 3.2.12, we showed that if **E** is **sPA1**$_\ell$ secure, then it is also **sPA1**$^+_\ell$ secure. As before, the high level idea behind the proof is to show that the security of cNM-CCA1 game can be reduced to the security of the primitives used in its construction. What makes our cNM-CCA1 construction unique is that there exists an extractor that can extract the plaintext of the ciphertexts which are submitted before revealing the challenge ciphertext with overwhelming probability. Therefore, using the extractor, there is no need to use the secret key to decrypt the ciphertexts which do not depend on the challenge ciphertext. Also we can use the extractor in the proofs as the substitution for the decryption oracle for the inner ciphertext of the *first* parallel query in the hybrids.

To prove that $\Pi^{(c)}$ is a cNM-CCA1 encryption scheme, we need to show that for any ppt adversary $\mathcal{A}$ and for all polynomials $p(.)$ and ppt distinguisher $\mathcal{D}$,

$$\left\{\text{(c)NME}_0\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\right\}_{k\in\mathbb{N}} \approx_c \left\{\text{(c)NME}_1\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\right\}_{k\in\mathbb{N}}$$

We show this by a hybrid argument. Consider the following experiments:

Experiment. $\text{cNME}_b{}^{(1)}(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$ acts exactly like cNME_b except for two changes. First, instead of selecting vkSig* when the challenge ciphertext is encrypted, choose this value as the first step of the experiment. Next, when processing decryption queries during the CCA1 phase and at the last step of the experiment, replace Verify with Verify* as follows:

Verify* Let vkSig* be the verification key in the challenge ciphertext $(c^*, \sigma^*, \text{vkSig}^*)$. Upon receiving a decryption query on $(c, \sigma, \text{vkSig})$, output $\perp$ if either vkSig $=$ vkSig* or $\text{Verify}_{\text{vkSig}}(c, \sigma) = 0$.

Experiment. $\mathsf{cNME}_b^{(2)}(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$ proceeds similar to $\mathsf{cNME}_b^{(1)}(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$ with the difference that the former experiment handles the decryption of the ciphertexts d_i's as follows (without loss of generality, assume that $d_i = (\vec{C}, \sigma, \text{vkSig})$):

NMDec$^*(\vec{C}, \sigma, \textbf{vkSig})$ Check if $1 = \text{Verify}^*_{\text{vkSig}}(\vec{C}, \sigma)$. For $i \in [k]$, do $C'_i \leftarrow \textbf{Dec}_{\text{SK}_i^{\text{vkSig}_i}}(C_i)$ (where SK_i's are the secret keys for the outer layer)

Check if $\forall_i, C'_1 = C'_i$. If all checks pass, use extractor $C^*_{\mathcal{A}}$ (defined below in Claim 7) to extract the plaintext of the inner ciphertext on C_i where i is the first spot where $\text{vkSig}_i \neq \text{vkSig}^*_i$ where vkSig^* is the verification key of the challenge ciphertext. Return the extracted plaintext.

Claim 6. *For* $b \in \{0,1\}$, $\left\{(\mathsf{c})\mathsf{NME}_b\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\right\} \approx_c \left\{(\mathsf{c})\mathsf{NME}_b^{(1)}\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\right\}$

Proof. Follows from the security of the signature scheme. □

Claim 7. *For* $b \in \{0,1\}$, $\left\{(\mathsf{c})\mathsf{NME}_b^{(1)}\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\right\} \approx_c \left\{(\mathsf{c})\mathsf{NME}_b^{(2)}\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\right\}$

Proof. The proof is very similar to the proof of Lemma 3.3.3 with the difference that here the inner layer consists encryptions under $(c-1)\cdot k + 1$ public keys (generated by calling **Gen** algorithm) compared to only 1 in the Π encryption scheme.

For completeness, we define $\mathbf{C}_{\mathcal{A}}$ and its corresponding extractor $\mathbf{C}^*_{\mathcal{A}}$ as in Lemma 3.3.3. In particular, we construct $\mathbf{C}_{\mathcal{A}}$ from $\mathcal{A}$ as follows:

- $\mathbf{C}_{\mathcal{A}}$ takes as input $2\cdot c\cdot k+1$ public keys $\left(\text{PK}_0, \{\text{PK}_i^b\}_{i\in[0\ldots ck], b\in\{0,1\}}\right)$. It generates a pair of signing keys $(\text{vkSig}^*, \text{skSig}^*) \leftarrow \mathsf{GenKey}(1^k)$ internally and sets $\text{PK} = \text{ReArrange}(\vec{\text{PK}}, \text{vkSig}^*)$ where the function ReArrange is presented in Figure 3.9. Intuitively the function ReArrange rearranges the keys given as input to it in a way that all of the input public keys $\{\text{PK}_i^0\}_{i\in[0\ldots ck], b\in\{0,1\}}$ are going to get into the positions consistent with the verification key such that later $\mathbf{C}_{\mathcal{A}}$ can sign a ciphertext encrypted under such keys as directed in $\Pi^{(c)}$.

  $\mathbf{C}_{\mathcal{A}}$ runs $\mathcal{A}_1$ on the input of PK. Note that this rearrangement of keys is crucial to make the view of the adversary $\mathcal{A}$ on the arrangement of the keys perfectly close to its view

ReArrange($\vec{\text{PK}}$, vkSig)
1: Parse $\vec{\text{PK}}$ as $\big(\text{PK}_0, \{\text{PK}_i^b\}_{i\in[0...ck],b\in\{0,1\}}\big)$
2: **for** $i \in [0..c-1]$
3: **for** $j \in [k]$
4: **if** $\text{vkSig}_j = 1$ **then** swap the values $\text{PK}^0_{i\cdot k+j}$ and $\text{PK}^1_{i\cdot k+j}$
5: **endFor**
6: **endFor**
7: Return $\text{NPK}^{(c)} = \big(\text{PK}_0, \{\text{PK}_i^b\}_{i\in[0...ck],b\in\{0,1\}}\big)$

Figure 3.9: THE DEFINITION FOR THE REARRANGE FUNCTION

in a real $\mathsf{cNME}_b{}^{(2)}(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment. For example, the adversary $\mathcal{A}$ might abort whenever the keys used in the outer layer of the challenge ciphertext have index $b = 0$ in $\text{PK} = \big(\text{PK}_0, \{\text{PK}_i^b\}_{i\in[0...ck],b\in\{0,1\}}\big)$. Such a coincidence occurs in the simulated $\mathsf{cNME}_b{}^{(2)}(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment with probability 1 if $\mathbf{C}_\mathcal{A}$ does not rearrange the public keys, while this coincidence occurs in a real $\mathsf{cNME}_b{}^{(2)}(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment with negligible probability due to the security of the signature scheme.

- Parse PK as $\{\text{NPK}^{(c-1)}, \{\text{PK}_i^b\}_{i\in[k],b\in\{0,1\}}\}$. Whenever $\mathbf{C}_\mathcal{A}$ receives a query $\big(\{y'_i\}_{i\in[k]}, \sigma, \text{vkSig}\big)$ from $\mathcal{A}_1$, it first checks if the signature is valid. If not, it returns $\perp$ as the answer to this query. Otherwise, it checks whether $\text{vkSig} = \text{vkSig}^*$. If so, it aborts. Otherwise, $\mathbf{C}_\mathcal{A}$ submits y'_i's one by one to the extractor to be extracted under $\text{PK}_i^{\text{vkSig}_i}$. If all of the queries do not get decrypted to the same value, $\mathbf{C}_\mathcal{A}$ returns $\perp$ to $\mathcal{A}_1$ as the answer to that query. Assuming all of the queries get decrypted to the same value y_0, $\mathbf{C}_\mathcal{A}$ then submits y_0, which is supposedly a ciphertext in the range of $\text{NMEnc}^{(c-1)}_{\text{NPK}^{(c-1)}}(.)$, to the extractor to be extracted recursively. $\mathbf{C}_\mathcal{A}$ then returns the result plaintext to $\mathcal{A}_1$. Eventually $\mathcal{A}_1$ returns (m_0, m_1, St) and halts.

- $\mathbf{C}_\mathcal{A}$ outputs (m_0, m_1), and accesses its oracle $\mathcal{O}$ for k times, getting in return $r_y = f^{-1}(\text{PK}_i^{\text{vkSig}_i^*}, y_i)$ on the i^{th} access. $\mathbf{C}_\mathcal{A}$ computes $y_i = f(\text{PK}_i^{\text{vkSig}_i^*}, r_y)$. $\mathbf{C}_\mathcal{A}$ then computes $\sigma^* = \mathsf{Sign}(\vec{y}, \text{skSig}^*)$, and runs $\mathcal{A}_2$ on the input $y^* = (\vec{y}, \sigma^*, \text{vkSig}^*)$ and St.

- Eventually $\mathcal{A}_2$ returns a vector of ciphertexts $\vec{Y}$ and the state information S. For all $j \in [|\vec{Y}|]$, $\mathbf{C}_\mathcal{A}$ does the following: on the query $Y_j = (\{y_i\}_{i\in[k]}, \sigma, \text{vkSig})$: it first checks if the signature

is valid. Otherwise, it checks whether vkSig $=$ vkSig*. If either of these two checks fail, the answer to this query will be $\perp$. Otherwise, $\mathbf{C}_{\mathcal{A}}$ finds the first index i where $\text{vkSig}_i \neq \text{vkSig}_i^*$, and submits y_i to its extractor to be extracted under $\text{PK}_i^{\text{vkSig}_i}$. $\mathbf{C}_{\mathcal{A}}$ then recursively submits the answer from the extractor (which is supposedly a ciphertext in the range of $\text{NMEnc}^{(c-1)}_{\text{NPK}^{c-1}}(.)$) again to the extractor to be decrypted. Call the answer from the extractor m'_j.

Eventually $\mathbf{C}_{\mathcal{A}}$ returns $\{Y_j, m'_j\}_{j\in[|\vec{Y}|]}$ and the state information S and halts.

As in the proof of Lemma 3.3.3, it is arguable that there exists an extractor $\mathbf{C}^*_{\mathcal{A}}$ for the ciphertext creator $\mathbf{C}_{\mathcal{A}}$ that can decrypt all the queries submitted by $\mathbf{C}_{\mathcal{A}}$ with overwhelming probability.

□

Notice that in the (c)$\text{NME}_b^{(2)}$ experiment, the extractor can only help to decrypt the first round of the adversary's parallel queries. We still need to figure out a way to decrypt the next $(c-1)$ rounds of the parallel queries. One candidate is to make an adversary $\mathcal{B}$ for the $(c-1)\text{NME}_b$ game that runs $\mathcal{A}$ internally and simulate the $\text{cNME}_b^{(2)}$ game for it. $\mathcal{B}$ would decrypt the first round of queries using extractor, and ask the outside to decrypt the next $(c-1)$ round of the queries. However there is a difficulty with this approach: in the beginning, it is not obvious which of the $2k$ keys will be chosen to encrypt the inner ciphertext in NME_b. However $\mathcal{B}$ has to specify the arrangement of the keys in the beginning for $\mathcal{A}$. Later when $\mathcal{B}$ receives the challenge ciphertext (along with its signature) from the NME_b game, $\mathcal{B}$ will need to add another layer of encryption on top of that challenge ciphertext which means that the signature of the challenge ciphertext needs to be changed. This requires forging the signature scheme which is impossible. One way to resolve this issue is to change the $\Pi^{(c)}$ scheme; instead of the outer layer, sign the inner layer and encrypt the signature along with the inner ciphertext. This way $\mathcal{B}$ will only need to add another layer of encryption to the challenge ciphertext (without knowing the signing key) and pass it to $\mathcal{A}$. However there are difficulties with this approach as well: the extractor works if it knows the signing keys, and we cannot assume it will work without such knowledge.

To get around this issue, we observe that in the proof that Π is a non-malleable CCA1 encryption scheme, we only used the signature scheme to ensure the set of keys in the outer layer of adversary's

queries are different from the set of keys of the outer layer in the challenge ciphertext. However, we can design a second encryption scheme in which always in the challenge ciphertext, a pre-chosen set of keys will be used, and the adversary cannot query the decryption oracle on that exact same set of keys. Since the decryption oracle works slightly differently and more restricted from a normal decryption oracle in (c)NME$_b$ game, we refer to this version of the (c)NME$_b$ experiment as the (c)NME$_b{}^*$ experiment, and the encryption scheme as $\Pi^{*(c)}$ which we define in claim 8. We prove that any adversary $\mathcal{A}$ has negligible advantage in distinguishing between (c)NME$_0{}^*$ and (c)NME$_1{}^*$. Later we show that cNME$_b{}^{(c)}(\Pi^{(c)}, \mathcal{A}, k, p(k)) \equiv (c-1)\text{NME}_b{}^*(\Pi^{*(c)}, \mathcal{B}, k, p(k))$.

Claim 8. *If the encryption scheme* $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *is weakly simulatable and* $\boldsymbol{sPA1}_\ell$ *secure, then for all ppt adversaries and distinguishers* $\mathcal{A}$ *and* $\mathcal{D}$ *respectively and for all constants* c *and polynomials* $p(.)$*:*

$$\left\{\text{(c)NME}_0{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k \approx_c \left\{\text{(c)NME}_1{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k$$

where experiment (c)NME* *is defined as follows:*

(c)NME$_b{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$

1: $(\text{NPK}^{(c)}, \text{CNSK}^{(c)}) \leftarrow \text{NMGen}^{*(c)}(1^k)$

2: $(m_0, m_1, S_1) \leftarrow A_0^{\text{NMDec}^{*(c)}(\text{NSK}^{(c)},.)}(\text{CNPK})$ s.t. $|m_0| = |m_1|$ and $\text{NMDec}^{*(c)}(\text{NSK}^{(c)}, Y = (y, \alpha))$ returns $\perp$ if $\alpha = 0^k$

3: $y \leftarrow \text{NMEnc}^{*(c)}(\text{NPK}^{(c)}, m_b)$

4: $\vec{d_1} \leftarrow \perp$

5: *for* $i = 1$ *to* c

6: $(\vec{C}, S_{i+1}) \leftarrow \mathcal{A}_i(y, S_i, \vec{d_i})$ where $|\vec{c}| = p(k)$

7: $d_{i+1,j} \leftarrow \text{NMDec}^{*(c)}(\text{NSK}^{(c)}, C_j = (c_j, \alpha))$ if $c_j \neq y$ and $\alpha \neq 0$ o.w. $d_{i+1,j} \leftarrow \perp$; $\forall j \in [|\vec{C}|]$

8: Output $\mathcal{D}(\vec{d}_{c+1}, S_{c+1})$

Figure 3.10: THE (c)NME* EXPERIMENT

Where the encryption scheme $\Pi^{*(c)} = (\mathit{NMGen}^{*(c)}, \mathit{NMEnc}^{*(c)}, \mathit{NMDec}^{*(c)})$ *is defined as follows:*

- $\mathit{NMGen}^{*(c)}(1^k)$ *is defined as* $\mathit{NMGen}^{(c)}(1^k)$ *in Figure 3.8,*

- *NMEnc*$^{*(c)}(\text{NPK}^{(c)}, m, \alpha \in \{0,1\}^k)$ *is defined as NMEnc*$^{(c)}((\text{NPK}^{(c)}, m, \alpha))$ *Figure 3.8,*
- *NMDec*$^{*(c)}(\text{NSK}^{(c)}, Y = (y, \alpha \in \{0,1\}^k))$ *is defined as NMDec*$^{(c)}(\text{NSK}^{(c)}, y, \alpha)$ *Figure 3.8.*

Proof. Instead of proving this claim directly, we only argue that $\Pi^{*(c-1)}$ is $(c-1)$NME* secure *if* the encryption scheme $\Pi^{(c-1)}$ described in Figure 3.8 is $(c-1)$NME secure. Notice that the latter is true by the inductive hypothesis.

To do so, we note that in the $(c-1)\mathsf{NME}_b{}^*$ experiment, the adversary cannot submit a query with $\alpha = 0^k$. This means that the adversary always has to submit queries that are encrypted under at least one new key compared to the challenge ciphertext. This is essentially posing the same restriction on the adversary as when an adversary wants to attack the $(c-1)$NME security of the encryption scheme Π^{c-1} (using at least one new key in the encryption of the outer layer of a ciphertext). In the latter case, because of unforgeability of the signature scheme, the adversary has to use at least one new key in its queries compared to the keys used in the challenge ciphertext. Notice that in the proof of Theorem 3.4.2 and Theorem 3.3.2 which claims that Π is a non-malleable CCA1 encryption scheme, we only use the signature scheme to enforce such a restriction. Therefore, very similarly to the proof for Theorem 3.4.2, it can be proven that $\Pi^{*(c-1)}$ is $(c-1)$NME* secure. □

Claim 9. *For any ppt adversary $\mathcal{A}$, polynomials $p(.)$ and security parameter k, there exists an adversary $\mathcal{B}$ s.t.*

$$(\mathsf{c})\mathsf{NME}_b{}^{(2)}(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k)) \equiv (c-1)\mathsf{NME}_b{}^*(\Pi^{*(c-1)}, \mathcal{B}, \mathcal{D}, k, p(k))$$

Proof. As discussed in claim 7, we can construct the ciphertext creator $\mathbf{C}_{\mathcal{A}}$ (that runs $\mathcal{A}$ internally) for which there exists an extractor $\mathbf{C}^*_{\mathcal{A}}$ that can decrypt the queries submitted by $\mathbf{C}_{\mathcal{A}}$ with overwhelming probability.

We build the $\mathsf{NME}_b{}^*$ adversary $\mathcal{B}$ that interacts with the $\mathsf{NME}_b{}^*$ experiment ("the outside"). Having the algorithm for $\mathcal{A}$, $\mathbf{C}_{\mathcal{A}}$ and $\mathbf{C}^*_{\mathcal{A}}$, the $(c-1)\mathsf{NME}_b{}^*$ adversary $\mathcal{B}$ acts as follows: $\mathcal{B}$ receives the public key $\text{NPK}^{*(c-1)} = (\vec{\text{PK}}' = \{\text{PK}_i'^b\}_{i\in[k],b\in\{0,1\}}, \text{PK}'_0)$ from the outside. and generates $2k$ keys as $(\text{PK}_i''^b, \text{SK}_i''^b) \leftarrow \mathbf{Gen}(1^k)$ for $i \in [k]$ and $b \in \{0,1\}$. Let $\vec{\text{PK}} = \left(\text{PK}'_0, \vec{\text{PK}}', \vec{\text{PK}}''\right)$ (this

arrangement of keys is necessary to make sure the keys that will be in the challenge ciphertext are in the right position). $\mathcal{B}$ runs $\mathbf{C}_{\mathcal{A}}$ (that simulates $\mathcal{A}$ internally) on $\vec{\text{PK}}$ and its random coins. Whenever $\mathbf{C}_{\mathcal{A}}$ asks a query, $\mathcal{B}$ runs $\mathbf{C}^*_{\mathcal{A}}$ to answer them ($\mathbf{C}^*_{\mathcal{A}}$ gets to know the random coins of $\mathbf{C}_{\mathcal{A}}$ and all its input as described in the proof of Lemma 3.3.3). Eventually $\mathbf{C}_{\mathcal{A}}$ outputs (m_0, m_1). $\mathcal{B}$ outputs m_0 and m_1 to the outside, and receives a ciphertext y. Remember that $\mathbf{C}_{\mathcal{A}}$ now accesses the oracle $\mathcal{O}$ for k times. Using fresh random coins for each encryption, $\mathcal{B}$ computes $C_i = \mathbf{Enc}_{\text{PK}_i''^0}(y)$ and sends $f^{-1}(\text{PK}_i''^0, C_i)$ to $\mathbf{C}_{\mathcal{A}}$ (and $\mathbf{C}^*_{\mathcal{A}}$) on the i^{th} access to $\mathcal{O}$. Eventually $\mathbf{C}_{\mathcal{A}}$ returns a vector of ciphertexts and their potential decryptions $\{Y_i, m'_i\}_{i\in[|\vec{Y}|]}$ and the state information S and halts. The only step left to do to determine the decryption of Y_i is to decrypt all the ciphertexts in the outer layer, and check that all of them get decrypted to the same value. $\mathcal{B}$ has all the $2k$ secret keys for the outer layer, hence it can do such a check. If the ciphertexts in the outer layer of Y_i do not decrypt to the same value, the decryption of Y_i would be $\perp$, otherwise the decryption for Y_i would be m'_i. After $\mathcal{B}$ decrypts all the Y_i, it submits the results along with the state information S back to $\mathcal{A}$ (notice that $\mathcal{A}$ is a (c)NME_b adversary and asks for c parallel queries). For another $c-1$ times, $\mathcal{A}$ returns a parallel query $\vec{Y'}$ and state information S'. Each time, $\mathcal{B}$ does the following for each ciphertext in $\vec{Y'}$: it checks all the outer layer ciphertexts are consistent using the secret keys for the outer layer. If there is any inconsistency in the outer layer of any of the ciphertexts, the decryption to that ciphertext would be $\perp$ otherwise $\mathcal{B}$ sets y'_i as the decryption of the first ciphertext in the outer layer of Y'_i. $\mathcal{B}$ then submits $\vec{y'}$ to the $(c-1)\mathsf{NME}_b{}^*$ game in the outside and receives the decryption to them. $\mathcal{B}$ then submits the decryptions along with the state information S' back to $\mathcal{A}$ in all the rounds but the last one. For the c^{th} query, $\mathcal{B}$ submits the decryption result and the state information to the distinguisher $\mathcal{D}$ and forwards $\mathcal{D}$'s output to the outside.

□

□

Chapter 4

Constructing CCA2-Secure Encryption from Weaker Encryption

4.1 Introduction

The standard security definition of an encryption scheme guarantees that the encryption of a message does not leak any information about the message aside from its length. In practice, this definition is too weak for network applications. Specifically, encryption schemes satisfying this definition are not necessarily secure when the adversary has access to a restricted decryption oracle. Depending on how restricted the decryption oracle is, different classes of security may be defined. In the least restricted and still meaningful form of decryption oracle access, the adversary may query the decryption oracle on anything but the challenge ciphertext. This notion of security is called adaptive chosen ciphertext (CCA2) security.

History Here, we follow Wee [31] by summarizing the background history of CCA2 constructions in the standard model. There have been two separate lines of work on the construction of CCA2 secure encryption schemes; one based on general assumptions starting from the work of Dolev, Dwork, Naor and Yung [9, 7, 19, 32, 33, 34, 35, 36, 28, 37] and another based on specific number-theoretic assumptions starting from the work of Cramer and Shoup [38, 39, 40, 41, 42, 43, 44, 45, 46]. The constructions based on specific assumption are generally more practical and efficient while the

constructions based on general assumptions are more flexible and may be instantiated based on different specific number-theoretic assumptions.

The CCA2 encryption schemes that are based on specific number-theoretic assumption may be categorized into encryption schemes based on decisional assumptions, e.g. Decisional Diffie-Hellman Assumption (DDH) or the quadratic residuosity assumption, and those based on computational assumptions, e.g. Computational Diffie-Hellman Assumption (CDH), finding shortest vectors in lattices or the factoring assumption. Generally, decisional assumptions are a much stronger class of assumptions than computational assumptions that are based on search problems. For instance, there are elliptic curve groups with a bilinear map in which the CDH assumption is believed to be hard, but the DDH assumption does not hold in them. As a result, the encryption schemes based on computational assumptions may be more secure and hence preferable to those based on decisional assumptions.

Canetti, Halevi and Katz [44] proposed the first practical CCA2 PKE based on a computational assumption called the Bilinear DH assumption in bilinear groups (BDH). Since then, several encryption schemes based on computational assumptions were proposed based on either CDH [45, 47, 48] or the factoring assumption [46]. However, none of these constructions provide a unifying framework to instantiate an encryption scheme based on different computational assumptions.

The first black-box constructions of CCA2 encryption schemes from general assumptions appeared in [36, 37] following the introduction of lossy trapdoor functions by Peikert and Waters in [35]. Prior work that used general assumptions required NIZK statements that were not black-box. The weakest general assumption in that line of work on which encryption schemes are based is (tag-based) adaptive trapdoor functions of Kiltz, Mohassel and O'Neil [37]. Although there is a black-box separation between adaptive trapdoor functions and lossy trapdoor functions [37, 49], the standard assumptions from which we can realize either of these general assumptions are not significantly different.

Another construction of a CCA2 encryption scheme based on general assumptions is that of Cramer, Hofheinz and Kiltz [50] which is based on the smooth projective hash proof system frame-

work introduced by Cramer and Shoup [40]. However, aside from CDH, the general assumptions on which their construction is based (hash proof systems and weaker than CCA2 encryption schemes) can only be instantiated from the RSA assumption which is presumably a stronger assumption than the factoring assumption.

The current state-of-the-art of construction of CCA2 encryption schemes from general assumptions is that of Wee [31] based on a primitive called the extractable hash proof system which is a special kind of non-interactive zero-knowledge proof of knowledge system. It is the first practical CCA2 encryption scheme from general assumptions that can be instantiated from both CDH and the factoring assumption. Given a hash proof system, the CCA2 encryption scheme can be constructed in a straightforward way. However the extractable hash proof system is specifically tailored to fit in their framework with quite a complicated and non-intuitive definition.

Our Result We build a provably CCA2 secure encryption scheme from any encryption scheme with two properties: the encryption should be secure under a one-way plaintext-checking attack (i.e. OW-PCA-secure) introduced in [16] and the scheme must support a "key malleability" property. We will formally define each of these notions later, but intuitively an encryption scheme is OW-PCA secure if given the public key and an encryption of a random message m, no ppt adversary can guess m correctly except with a negligible probability even with access to an oracle that takes a ciphertext and a message and verifies whether the ciphertext decrypts to the given message or not. This notion is seemingly weak: the oracle only provides "yes/no" answers about whether a decryption is correct, and the adversary must recover the entire message, and not just any single predicate as per semantic security.

An encryption scheme is *key malleable* if given a randomly generated public key PK_1 (which has secret key SK_1), there exists a ppt algorithm that can generate a new random looking public key PK_2 and trapdoor information s. Also, given SK_1 and s, there exists a ppt algorithm to generate a valid secret key SK_2 for the public key PK_2 such that $(\mathrm{PK}_1, \mathrm{SK}_1, \mathrm{PK}_2, \mathrm{SK}_2)$ is indistinguishable from a pair of freshly generated keys. Moreover, by using the trapdoor information s, there exists a method to transform a ciphertext encrypted under one of the public keys to a ciphertext encrypted

under the other, or vice-versa.

We use an encryption scheme that is OW-PCA secure and key malleable to construct an encryption scheme that is one-bit CCA2 secure (i.e. the ciphertext encrypts a single bit) using techniques similar to DDN-Lite with slight differences. Since Myers and shelat [17] showed that one-bit CCA2 security implies many-bit CCA2 security, we conclude that many-bit CCA2 security may be achieved using an encryption scheme that is only OW-PCA secure and key malleable.

The Main Technique Notice that because the OW-PCA scheme is essentially a one-way function, therefore there exists a hard-core function such that any ppt adversary cannot guess the hard-core bit of the encrypted message with advantage better than $1/2$. We instantiate the DDN-Lite construction with our OW-PCA scheme, and prove that an adversary that breaks the CCA2-security of the scheme can be used to guess the hard-core bit of the OW-PCA scheme with a probability better than $1/2$. Now let us describe how we simulate the CCA2 game for an adversary. Given an OW-PCA key pk_0, we use the key malleability property to generate several other keys used in the DDN-construction (and we save their trapdoor information for later). We choose a special signing key skSig* and place the pk_0 key at one of the spots in which that key will be used in skSig* and fill for the rest of the spots in skSig* using public keys that were mauled from PK_0. For the spots that are not in skSig*, we generate fresh pairs of public and secret keys.

Given the challenge ciphertext c^* from the OW-PCA experiment which is encrypted under PK_0, we use the trapdoor information to maul it to ciphertexts for the other spots in skSig*, sign all of them and send them to the adversary.

In order to answer decryption queries from the adversary, we first argue that by the security of the signature scheme, it is guaranteed that at least one key in the query was not used in the challenge ciphertext if the signature is valid. Thus, we can decrypt ciphertexts in the query for which our simulation knows the secret key for and verify that all of these ciphertexts decrypts to the same value which we call m. For the spots that we do not have the secret key for, we have the trapdoor information to maul those ciphertext into ciphertexts that are encrypted under PK_0. Then we submit all the mauled ciphertext and m one by one to the "plaintext checking oracle" oracle. If any of

the verifications so far failed, we return $\perp$ as the response to the query, otherwise we return the hard-core bit of m. Eventually the adversary returns a bit b and halts. We forward that bit to the OW-PCA experiment as our guess of the hard-core bit of the message that was encrypted in c^*.

Contribution The approach that we put forward for constructing CCA2-secure encryption only requires an encryption scheme as a starting primitive. In other words, we start from a trapdoor predicate (albeit, with a few notable properties). As mentioned above, prior work on constructing CCA2-secure encryption all require complexity assumptions that seem stronger than a *predicate*. Namely, these prior works require either trapdoor permutations, lossy trapdoor functions, a more complicated smooth projective hash system, or a more complicated extractable hash proof system. In these latter two cases, the hash proof system requires discussion of various modes, and several auxiliary functions.

Although our constructions use an adaptive complexity assumption (i.e, an assumption that itself contains an oracle), we note that any construction of CCA2-secure encryption from say—CCA1-secure encryption—would represent serious progress in this research area. Thus, the use of an oracle in our assumption does not seem on face unreasonable. Another way to view our assumption is with respective to the work of Myers and shelat who show that one-bit CCA2-secure encryption implies many-bit CCA2-encryption. In analogue, we show that a scheme that is secure in the presence of an oracle that only reveals one-bit about a ciphertext (namely, whether it is a proper encoding of a given message) can be transformed into a scheme that is secure with a full CCA2 oracle. Although we do not quite achieve this result (since our construction requires these additional key-malleability properties), we view our work as introducing a new conceptual approach to CCA2 encryption.

To show the viability of our approach, we show that our security notion can be instantiated with the standard number theoretic complexity assumptions. We conjecture that LWE-based complexity assumptions can also be used to construct a scheme that will satisfy our two required properties (indeed, LWE schemes seem to naturally support the key homomorphic properties).

4.2 Preliminaries and Definitions

We consider the definition of one-way plaintext checking attack secure encryption schemes as given by Okamote and Pointcheval [16].

Definition 4.2.1 (OW-PCA). *[16] We say that* $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *with the family of plaintext spaces* $\{\mathcal{M}_k\}_{k\in\mathbb{N}}$ *where* $|\mathcal{M}_k| \in \omega(\log k)$ *is one-way plaintext checking attack secure (*OW-PCA *secure) if for all ppt adversaries* $\mathcal{A}$ *and* $k \in \mathbb{N}$*, we have that* $\Pr[\mathsf{OW\text{-}PCA}(\boldsymbol{E}, \mathcal{A}, k) = 1] < \epsilon(k)$*. The* OW-PCA *experiment is defined as follows:*

$\mathsf{OW\text{-}PCA}(\mathbf{E}, \mathcal{A}, k)$
1: $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathbf{Gen}(1^k)$
2: $m \leftarrow \mathcal{M}_k$
3: $c \leftarrow \mathbf{Enc}_{\mathrm{PK}}(m)$
4: $m' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{SK}}(\cdot,\cdot)}(\mathrm{PK}, c)$
5: Output $m \stackrel{?}{=} m'$ where $\mathcal{O}_{\mathrm{SK}}(c, m)$ returns $\top$ if $m = \mathbf{Dec}_{\mathrm{SK}}(c)$ and $\perp$ otherwise.

Figure 4.1: THE OW-PCA EXPERIMENT

Observe that the adversary only wins the game if the entire message is recovered, and therefore no particular bit of **Enc** is hidden. In this sense a OW-PCA secure resembles a strengthening of a trapdoor function, more than an encryption scheme. We are interested in one of the hard-core bits of the family of encryption functions $\{\mathbf{Enc}_{\mathrm{PK}}\}_{(pk,sk)\leftarrow g(1^k)}$, that is the same hard-core bit is shared over all of the public-keys of the same security parameter. Such bits are known to exist inherently for many hardness assumptions, or can be constructed through the Goldreich-Levin transformation [51]. Let the function h be a hard-core predicate function with respect to an encryption function $\mathbf{Enc}_{\mathrm{PK}}$ for the message space $\mathcal{M}_k$, for the OW-PCA secure encryption scheme $\mathbf{E}$. We define the experiment $\mathsf{OW\text{-}PCA}_b$ as the predicate guessing version of OW-PCA. That is $\mathsf{OW\text{-}PCA}_b(\mathbf{E}, \mathcal{A}, k)$ is the same way as $\mathsf{OW\text{-}PCA}(\mathbf{E}, \mathcal{A}, k)$, but where i) m is chosen randomly conditioned on $h(m) = b$; and ii) the adversary returns a bit b' as its guess of $h(m)$ and the experiment returns 1 iff $b = b'$. Clearly $\Pr[\mathsf{OW\text{-}PCA}_b(\mathbf{E}, \mathcal{A}, k) = 1] < 1/2 + \epsilon(k)$.

4.3 Indistinguishable Key Generation, Inter-key Ciphertext Malleability and Ciphertext Rerandomization

In this section we consider three required auxiliary properties of OW-PCA schemes. The first property of the scheme is that given a public-key in the scheme we are able to generate alternate public-keys for the scheme that come from a distribution that is indistinguishable from the original key-generation algorithm. The second property is that when we generate an alternate public-key PK′ based on an original public-key PK, it is possible to generate some state information so that we can similarly transform encryptions c under PK into encryptions c' under PK′, and vice-versa. Finally, we require that the ciphertexts are re-randomizable. In Sections 4.5 and 4.6, we show natural constructions of OW-PCA schemes which have these auxiliary properties.

Definition 4.3.1. *[Alternate Keys and Inter-Key Ciphertext Malleable]*

Consider $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *with the plaintext space* $\mathcal{M}$ *and the key space* $\mathsf{K} = \{\mathsf{K}_n\}_{n\in\mathbb{N}}$ *which is a family of finite sets of public and secret keys* (PK, SK)*. For ease of notation we assume that the secret key, the random coins and the message in the encryption scheme* $\boldsymbol{E}$ *is* k *where* k *is the security parameter. We say* $\boldsymbol{E}$ *is key malleable if there exists probabilistic poly-time algorithms* $\vec{f} = (f, f', f_1, f_2, f_r)$ *such that for all security parameter* $k \in \mathbb{N}$ *the following holds:*

1. *(***Alternate Indistinguishable Keys***) (a) We call the first distribution* **Dist** *and the second distribution* **Dist'**.

$$\{(PK_1, SK_1) \leftarrow \boldsymbol{Gen}(1^k), (PK_2, SK_2) \leftarrow \boldsymbol{Gen}(1^k) : (PK_1, SK_1, PK_2, SK_2)\}_{k\in\mathbb{N}} \approx_c$$
$$\{(PK_1, SK_1) \leftarrow \boldsymbol{Gen}(1^k), (PK_2, s_2) \leftarrow f(PK_1), SK_2 \leftarrow f'(SK_1, s_2) : (PK_1, SK_1, PK_2, SK_2)\}_{k\in\mathbb{N}}$$

(b) Also there is a negligible function ϵ *such that the following holds:*

$$\Pr\left[(PK_1, SK_1) \leftarrow \boldsymbol{Gen}(1^k), (PK_2, s_2) \leftarrow f(PK_1), SK_2 \leftarrow f'(SK_1, s_2) : (PK_2, SK_2) \in K_k\right] \geq 1 - \epsilon(k)$$

2. *(***Inter-Key Ciphertext Malleability***) For all sufficiently large k, for all $m \in \mathcal{M}_k$, there is a negligible function ϵ it holds:*

$$\Pr\left[(PK, SK) \leftarrow \textbf{\textit{Gen}}(1^k), (PK', s) \leftarrow f(pk) : \left(\begin{array}{l} \textbf{\textit{Enc}}_{PK}(m) \equiv_s f_1(s, \textbf{\textit{Enc}}_{PK'}(m)) \wedge \\ \textbf{\textit{Enc}}_{PK'}(m) \equiv_s f_2(s, \textbf{\textit{Enc}}_{PK}(m)) \end{array}\right)\right] \geq 1-\epsilon(k)$$

3. *(***Ciphertext ReRandomization***) We can rerandomize ciphertexts under a given public-key. Namely, for every sequence of public- and private-key pairs, messages in corresponding message spaces and corresponding ciphertexts:* $S = \{(PK, m, c)\}_{k\in\mathbb{N},(pk,SK)\leftarrow g(1^k),m\in\mathcal{M}_k,c\leftarrow \textbf{\textit{Enc}}_{PK}(m)}$:

$$\{f_r(PK, c)\}_S \approx_s \{\textbf{\textit{Enc}}_{PK}(m)\}_S.$$

4.4 CCA2 Security from Weaker Assumptions

In Figure 4.2 we give the construction of encryption scheme Π that is CCA2 secure if **E** is OW-PCA secure and has the properties mentioned in the previous section (i.e., Alternate Indistinguishable Public-Keys and Inter-Key Ciphertext Malleability). We present a brief intuitive description of the security reduction. At a high level, any adversary A that breaks the CCA2 security of the construction can be used to break the OW-PCA security of **E**. First, the construction uses multiple redundant encryptions under redundant public-keys from the **E** scheme. The use of non-duplicatable set selection is used (in the, by now, standard method proposed by Dolev et al.[9]) so that each legitimate query to a decryption oracle must use a different subset of the public-keys to perform the redundant encryptions. Now, if an adversary A breaks the CCA2 security, then we can construct an adversary A' for **E** that simulates A. This is done by simulating A, but giving it a public-key in which the public-keys used to create the challenge ciphertext for A, are not legitimate keys, but rather alternate indistinguishable keys, generated from A''s OW-PCA public-key, A' generates the remainder of the public-keys appropriately, and knows the corresponding secret-keys. In order to simulate A's decryption oracle, we use the fact that due to the unduplicatable set selection, any

decryption query will involve at least one public-key in the OW-PCA scheme to which A' knows the corresponding secret-key. A' decrypts all the ciphertexts for which it has the corresponding secret-keys, and ensures they all decrypt to a consistent value m. Similarly, it uses it owns validity oracle in coordination with the inter-key ciphertext malleability for the ciphertexts that are encrypted under public-keys for which A' does not know the corresponding secret-key to ensure that all of these ciphertexts are also encryptions of m.

We know give the formal statement of the theorem, and its proof.

Let $\mathbf{E} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ be any encryption scheme that is OW-PCA secure and Indistinguishable Key Generation, and inter-key ciphertext malleable with the message space $\{\mathcal{M}_k\}_{k\in\mathbb{N}}$, where $|\mathcal{M}_k| \in \omega(\log k)$, and a corresponding hard-core predicate $h(\cdot)$. Let $\Sigma = (\mathsf{GenKey}, \mathsf{Sign}, \mathsf{Verify})$ be a strong one-time signature scheme.

$\mathsf{gen}(1^k)$
1: $(\mathrm{PK}_i^b, \mathrm{SK}_i^b) \leftarrow \mathbf{Gen}(1^k),\ \forall i \in [k]$ and $b \in \{0,1\}$
2: Output $\mathrm{PK} = \{(\mathrm{PK}_i^b)\}_{i\in[k],b\in\{0,1\}}$ and $\mathrm{SK} = \{(\mathrm{SK}_i^b)\}_{i\in[k],b\in\{0,1\}}$

$\mathsf{enc}(\mathrm{PK}, b)$
1: $m \leftarrow \mathcal{M}_k$ *s.t.* $h(m) = b$
2: $(\mathsf{SigSK}, \mathsf{SigVK}) \leftarrow \mathsf{GenKey}(1^k)$
3: $(c_i \leftarrow \mathbf{Enc}_{\mathrm{PK}_i^{\mathsf{SigVK}_i}}(m))_{i\in[k]}$
4: $\sigma \leftarrow \mathsf{Sign}_{\mathsf{SigSK}}(\vec{c})$
5: Output $(\vec{c}, \mathsf{SigVK}, \sigma)$

$\mathsf{dec}(\mathrm{SK}, C = (\vec{c}, \mathsf{SigVK}, \sigma))$
1: **if** $\mathsf{Verify}_{\mathsf{SigVK}}(\sigma, \vec{c}) = 0$ **then** Output $\perp$
2: $(m_i' \leftarrow \mathbf{Dec}_{\mathrm{SK}_i^{\mathsf{SigVK}_i}}(c_i))_{i\in[k]}$
3: **if** $\exists i$ s.t. $(m_1' \neq m_i')_{i\in[k]}$ **then** Output $\perp$
4: Output $h(m_1')$

Figure 4.2: THE PROPOSED CCA2 SECURE ENCRYPTION SCHEME Π

Theorem 4.4.1. *The encryption scheme Π presented in Figure 4.2 is* CCA2 *secure if* ***E*** *is* OW-PCA *secure and Indistinguishable Key Generation, and inter-key ciphertext malleable with the message space $\{\mathcal{M}_k\}_{k\in\mathbb{N}}$, where $|\mathcal{M}_k| \in \omega(\log k)$, and h is a corresponding hard-core predicate. Let $\Sigma = (\mathsf{GenKey}, \mathsf{Sign}, \mathsf{Verify})$ be a strong one-time signature scheme.*

Proof. To prove that Π is $\mathsf{CCA2}$ secure, we need to show that for any ppt adversary $\mathcal{A}$,

$$\{\mathsf{CCA2}_0(\Pi, \mathcal{A}, k)\}_{k\in\mathbb{N}} \approx_c \{\mathsf{CCA2}_1(\Pi, \mathcal{A}, k)\}_{k\in\mathbb{N}}$$

We show this by a series of hybrid arguments. Consider the following experiments:

Experiment $\mathsf{CCA2}_b{}^{(1)}(\Pi, \mathcal{A}, k)$. It modifies $\mathsf{CCA2}_b$ in two ways. First, instead of selecting vkSig* when the challenge ciphertext is encrypted, choose this value as the first step of the experiment. Second, when processing decryption queries during the experiment, replace Verify with Verify* as follows:

Verify* Let vkSig* be the verification key in the challenge ciphertext $(\vec{c^*}, \sigma^*, \text{vkSig}^*)$. Upon receiving a decryption query on $(\vec{c}, \sigma, \text{vkSig})$, output $\perp$ if either vkSig $=$ vkSig* or $\text{Verify}_{\text{vkSig}}(\vec{c}, \sigma) = 0$.

Claim 10. *For* $b \in \{0, 1\}$, $\{\mathsf{CCA2}_b\,(\Pi, \mathcal{A}, k)\}_{k\in\mathbb{N}} \approx_c \{\mathsf{CCA2}_b{}^{(1)}\,(\Pi, \mathcal{A}, k)\}_{k\in\mathbb{N}}$

Proof. Follows from the security of strongly secure signature scheme using standard techniques. □

For the sake of simplicity of the presentation, in the rest of the proof we assume vkSig$^* = 0^k$. Our proof generalizes in the normal ways to handle any particular signature string.

Experiment $\mathsf{CCA2}_b{}^{(2)}(\Pi, \mathcal{A}, \vec{f}, k)$. We define k hybrids $\{H_i\}_{i\in[k]}$ where H_1 is defined exactly as $\mathsf{CCA2}_b{}^{(1)}\,(\Pi, \mathcal{A}, k)$. In each hybrid, we generate more of the public-key and challenge ciphertext using the extra key and malleability properties of the $\mathsf{OW\text{-}PCA}$ secure scheme. Hybrid H_i, for $2 \le i \le k$, modifies H_{i-1} in the following ways:

1. Instead of calling $\mathbf{Gen}(k)$ to generate the key (PK_i^0), call the function $(\text{PK}_i^0, s_i) \leftarrow f(\text{PK}_1^0)$ using fresh and independent random coins.

2. Generate the i^{th} element in $\vec{c^*}$, representing an encryption of m_b, in the challenge ciphertext by calling $f_1(s_i, c_1^*)$ where $c_1^* = \mathbf{Enc}_{\text{PK}_1^0}(m_b)$ (instead of $\mathbf{Enc}_{\text{PK}_i^0}(m)$).

3. After revealing the challenge ciphertext, the adversary would ask queries of the form $q = (\vec{c}, \mathsf{SigVK}, \sigma)$. H_i decrypts the query as in H_{i-1} as before but with the difference that the decryption of c_i is computed as $\mathbf{Dec}_{\mathrm{SK}_1^0}(f_2(s_i, c_i))$ (instead of $\mathbf{Dec}_{\mathrm{SK}_i^0}(c_i)$).

Define $\mathsf{CCA2}_b^{(2)}(\Pi, \mathcal{A}, \vec{f}, k)$ equivalent to H_k.

Claim 11. *$H_{i-1} \approx_c H_i$ for $2 \leq i \leq k$.*

Proof. Notice that due to the **Inter-Key Ciphertext Malleability** property (see Definition 4.3.1), with overwhelming probability, $\mathbf{Enc}_{\mathrm{PK}_1^0}(m) \equiv f_2(s_i, \mathbf{Enc}_{\mathrm{PK}_i^0}(m))$ and $\mathbf{Enc}_{\mathrm{PK}_i^0}(m) \equiv f_1(s_i, \mathbf{Enc}_{\mathrm{PK}_1^0}(m)$ for any message $m \in \mathcal{M}_k$.

Assume there exists an index $2 \leq i \leq k$ and ppt adversary $\mathcal{A}$ such that there exists a ppt distinguisher $\mathcal{D}$ that can distinguish the two hybrids. Then we build a ppt distinguisher $\mathcal{D}'$ for distinguishing between **Dist** and **Dist'** (see Definiton 4.3.1) with polynomial probability.

The distinguisher $\mathcal{D}'$ receives $(\mathrm{PK}_1, \mathrm{SK}_1, \mathrm{PK}_2, \mathrm{SK}_2)$ as input. It simulates $\mathcal{A}$. $\mathcal{D}'$ computes $(\mathrm{PK}'_j, s'_j) \leftarrow f(\mathrm{PK}_1)$ for $2 \leq j \leq i-1$ and $(\mathrm{PK}'_j, \mathrm{SK}'_j) \leftarrow \mathbf{Gen}(k)$ for $i+1 \leq j \leq k$ and $(\mathrm{PK}''_j, \mathrm{SK}''_j) \leftarrow \mathbf{Gen}(k)$ for $j \in [k]$. $\mathcal{D}'$ then sets the public key PK given to its simulation of $\mathcal{A}$ as follows:

$$\text{for } j \in [0 \ldots k] \;\&\; \alpha \in \{0,1\}, \mathrm{PK}_i^\alpha = \begin{cases} \mathrm{PK}_1 & \text{if } j = 1 \text{ and } \alpha = 0 \\ \mathrm{PK}_2 & \text{else if } j = i \text{ and } \alpha = 0 \\ \mathrm{PK}'_j & \text{else if } \alpha = 0 \\ \mathrm{PK}''_j & \text{otherwise} \end{cases}$$

$\mathcal{D}'$ then picks a random bit b and computes $C^* = \mathsf{enc}(\mathrm{PK}, b)$ (using SigSK^*) and sends PK along with C^* to the simulation of $\mathcal{A}$ when it requests the challenge ciphertext. Whenever $\mathcal{A}$ asks a query $C = (\vec{c}, \mathsf{SigVK}, \sigma)$ to the decryption oracle, $\mathcal{D}'$ computes the decryption exactly as dec algorithm with the difference that on ciphertexts encrypted under the indistinguishable keys generated from PK_1, it uses the corresponding state information s_j and computes the decryption as $\mathbf{Dec}_{\mathrm{SK}_1^0}(f_2(s_j, c_j))$ (instead of $\mathbf{Dec}_{\mathrm{SK}_j^0}(c_j)$) which are guaranteed to result in the same values because

of the **Inter-Key Ciphertext Malleability** property (see Definition 4.3.1) and that $(\text{PK}_2, \text{SK}_2)$ are valid keys according to the part (b) of the **Alternate Indistinguishable Keys** property. Eventually $\mathcal{A}$ returns a bit b' and halts. Notice that when $(\text{PK}_1, \text{PK}_2) \in$ **Dist**, H_{i-1} is simulated and when $(\text{PK}_1, \text{PK}_2) \in$ **Dist'**, H_i is simulated with overwhelming probability (due to the slight difference in how $\mathcal{D}'$ creates the challenge ciphertext and answers the queries which is guaranteed to produce the same result because of the **Inter-Key Ciphertext Malleability** property and part (b) of the **Alternate Indistinguishable Keys** property in Definition 4.3.1). $\mathcal{D}'$ then runs $\mathcal{D}$ on the view of the adversary, and returns 0 to the outside if $\mathcal{D}$ outputs H_{i-1} and 1 otherwise. □

Claim 12. *For* $b \in \{0,1\}$, $\{\mathsf{CCA2}_b{}^{(1)}(\Pi, \mathcal{A}, k)\}_{k\in\mathbb{N}} \approx_c \{\mathsf{CCA2}_b{}^{(2)}\left(\Pi, \mathcal{A}, \vec{f}, k\right)\}_{k\in\mathbb{N}}$

Proof. In Claim 11 we proved that $H_{i-1} \approx_c H_i$ for $2 \leq i \leq k$. Since H_1 is $\mathsf{CCA2}_b{}^{(1)}(\Pi, \mathcal{A}, k)$ and H_k is $\mathsf{CCA2}_b{}^{(2)}(\Pi, \mathcal{A}, \vec{f}, k)$, the claim follows from the fact that there is a polynomial chain of computationally indistinguishable distributions that separate $\mathsf{CCA2}_b{}^{(1)}(\Pi, \mathcal{A}, k)$ and $\mathsf{CCA2}_b{}^{(2)}(\Pi, \mathcal{A}, \vec{f}, k)$, implying they are themselves computationally indistinguishable. □

Claim 13. *For every ppt adversary* $\mathcal{A}$*, there exists a ppt adversary* $\mathcal{B}$ *such that for* $b \in \{0,1\}$*,*

$$\{\mathsf{CCA2}_b{}^{(2)}(\Pi, \mathcal{A}, \vec{f}, k)\}_{k\in\mathbb{N}} \approx_s \{\mathsf{OW\text{-}PCA}_b(\boldsymbol{E}, \mathcal{B}, k)\}_{k\in\mathbb{N}}$$

Proof. We build the $\mathsf{OW\text{-}PCA}_b$ adversary $\mathcal{B}$ that simulates the adversary $\mathcal{A}$ in the $\mathsf{CCA2}_b{}^{(2)}$ experiment. $\mathcal{B}$ receives as input public key PK_1 and a ciphertext c_1. $\mathcal{B}$ uses Σ to create a signing- and verification-key (SigSK and SigVK) for the challenge ciphertext it will create in the $\mathsf{CCA2}_b{}^{(2)}$ simulation it performs. Again to ease presentation and notation, we assume the verification key $\mathsf{SigVK}^* = 0^k$. $\mathcal{B}$ computes $(\text{PK}'_j, s'_j) \leftarrow f(\text{PK}_1)$ for $2 \leq j \leq k$ and $(\text{PK}''_j, \text{SK}''_j) \leftarrow$ **Gen**(k) for $j \in [k]$. $\mathcal{B}$ then sets the public key PK given to $\mathcal{A}$ as follows:

$$\text{for } i \in [0 \ldots k] \;\&\; \alpha \in \{0,1\}, \text{PK}_i^\alpha = \begin{cases} \text{PK}_1 & \text{if } i = 1 \text{ and } \alpha = 0 \\ \text{PK}'_i & \text{else if } \alpha = 0 \\ \text{PK}''_i & \text{otherwise} \end{cases}$$

□

$\mathcal{B}$ prepares the challenge ciphertext C^* using

$$\vec{c^*} = (c_1, f_r(\mathrm{PK}_2^0, f_1(s_2', c_1)), f_r(\mathrm{PK}_3^0, f_1(s_3', c_1)), \ldots, f_r(\mathrm{PK}_k^0, f_1(s_k', c_1))$$

and signs using SigSK^*. $\mathcal{B}$ simulates $\mathcal{A}$ on inputs PK and C^*. Note that the distribution on the prepared $\vec{c^*}$ is statistically close to the proper distribution due to the properties of f_1 for transforming encryptions under PK_1 into encryptions under PK_i and f_r for its ability to rerandomize the encryptions, and make the resulting ciphertexts uncorrelated to c_1 (or each other). This is the only part of the simulation that is not perfect. Whenever $\mathcal{A}$ asks a decryption query $C = (\vec{c}, \mathsf{SigVK}, \sigma)$, $\mathcal{B}$ responds as follows, under the caveat that when a validity check fails during the simulation described, $\mathcal{B}$ returns $\perp$ as the answer to the corresponding query: $\mathcal{B}$ first checks that the verification-key is not the same as that of the challenge ciphertext in the simulation: $\mathsf{SigVK} \neq \mathsf{SigVK}^*$ (remember that for simplicity we assumed that $\mathsf{SigVK}^* = 0^k$), and next that the signature is valid. Then for the OW-PCA secure $\mathbf{E}$ ciphertexts embedded in C that $\mathcal{B}$ has the secret key for, it checks if they all get decrypted to the same value m_C. For each embedded ciphertext c_i for which $\mathcal{B}$ does not have the secret key, it computes $c' = f_2(s_i', c_i)$ and submits (c', m_C) to its PCA oracle to verify that c' is a valid encryption of m_C (notice that when $i = 1$, it just submits (c_i, m) to the oracle because it is encrypted under the same key as the outside). If it passes all checks, $\mathcal{B}$ forwards $h(m)$ to $\mathcal{A}$. Eventually $\mathcal{A}$ returns a bit b' and halts. $\mathcal{B}$ outputs b' and halts.

□

Putting all the hybrid experiments together shows that the proposed encryption scheme Π is CCA2 secure, as required by Theorem 4.4.1.

4.5 Encryption Based on the CDH Assumption

In this and the next section we present constructions of OW-PCA secure encryption schemes that have the alternate-key generation properties and the inter-ciphertext malleability property. Thus

these constructions can be applied directly to the previous section's construction to achieve CCA2 security.

Consider a cyclic group $\mathbb{G}$ with the generator g and of prime order q, using multiplicative notation. We define Computational Diffie-Hellman assumption, the Strong Diffie-Hellman assumption [52], the Twin Diffie-Hellman assumption [45], and the Strong Twin Diffie-Hellman assumption [45] as follows. The notation is mostly taken from [45]. All of these assumptions are known to be computationally equivalent.

The Computational Diffie-Hellman Assumption. For $X, Y, Z \in \mathbb{G}$, define

$$\text{dh}(X, Y) = Z, \textit{ where } X = g^x, Y = g^y, Z = g^{xy}$$

The Computational Diffie-Hellman (CDH) assumption holds for the group $\mathbb{G}$ if it is computationally intractable for an adversary in probabilistic polynomial time to compute $\text{dh}(X, Y)$ for random $X, Y \in \mathbb{G}$.

The Strong Diffie-Hellman Assumption. [52] For $X, \hat{Y}, \hat{Z} \in \mathbb{G}$, define

$$\text{dhp}(X, \hat{Y}, \hat{Z}) = \text{dh}(X, \hat{Y}) \stackrel{?}{=} \hat{Z}$$

The Strong Diffie-Hellman assumption (SDH) augments the adversary in the CDH assumption with an oracle that confirms, for a fixed x, if a triple is of the form (g^x, g^y, g^{xy}). That is at states that it is computationally intractable for an adversary to compute $\text{dh}(X, Y)$ for random $X, Y \in \mathbb{G}$ even with access to the decision oracle $\text{dhp}(X, \cdot, \cdot)$.

The Twin Diffie-Hellman Assumption. [45] For $X_1, X_2, Y \in \mathbb{G}$, define

$$\text{2dh}(X_1, X_2, Y) = (\text{dh}(X_1, Y), \text{dh}(X_2, Y))$$

The Twin Diffie-Hellman assumption states that it is computationally intractable to compute $\text{2dh}(X_1, X_2, Y)$ for random $X_1, X_2, Y \in \mathbb{G}$. In essence, one wants to compute two Diffie-Hellman

triples, which share the element g^y in common.

The Strong Twin Diffie-Hellman Assumption. [45] For $X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2 \in \mathbb{G}$, define

$$\text{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) = \text{2dh}(X_1, X_2, \hat{Y}) \stackrel{?}{=} \hat{Z}_1, \hat{Z}_2$$

The Strong Twin Diffie-Hellman assumption states that it is computationally intractable to compute $\text{2dh}(X_1, X_2, Y)$ for random $X_1, X_2, Y \in \mathbb{G}$ even with access to a decision oracle $\text{2dhp}(X_1, X_2, \cdot, \cdot, \cdot)$.

The following lemma establishes the equivalence of all of these computational hardness assumptions.

Lemma 4.5.1 (Theorem 1 in [45])**.** *The Computational DH assumption holds if and only if the strong twin DH assumption holds.*

We assume on inputs corresponding to security parameter k, there is a public parameter $\mathsf{PP} = (\mathbb{G}, g)$ denoting a cyclic group $\mathbb{G}$ with $\approx 2^k$ elements in it, and generator g that all algorithms have access to as input, but which is not denoted for clarity of description. The plaintext space $\mathcal{M}$ is composed of pairs of group elements.

Gen(1^k)
1: $x_1, x_2 \leftarrow \{0, 1, ..., |\mathbb{G}| - 1\}$
2: $X_i \leftarrow g^{x_i}$ for $i \in \{1, 2\}$
3: Output PK $= (X_1, X_2, g)$, SK $= (x_1, x_2)$

Enc$(\text{PK} = (X_1, X_2, g), m = (m_1, m_2))$
1: $r \leftarrow \{0, 1, ..., |\mathbb{G}| - 1\}$
2: Output $(m_1 \cdot X_1^r, m_2 \cdot X_2^r, g^r)$

Dec$(\text{SK} = (x_1, x_2), C = (c_1, c_2, R))$
1: $Y_i \leftarrow R^{x_i}$ for $i \in \{1, 2\}$
2: $m_i \leftarrow c_i / Y_i$ for $i \in \{1, 2\}$
3: Output $m = (m_1, m_2)$

Figure 4.3: THE ENCRYPTION SCHEME $\mathbf{E} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$

Claim 14. *The encryption scheme presented in 4.3 is* OW-PCA *secure if the strong twin DH assumption holds.*

Proof. For the sake of contradiction, assume there exists a ppt adversary $\mathcal{A}$ that breaks the OW-PCA security of $\mathbf{E}$ with the plaintext space $\mathcal{M}$. We build a ppt adversary $\mathcal{B}$ that breaks the strong twin DH assumption. The reduction works as follows. $\mathcal{B}$ simulates $\mathcal{A}$. $\mathcal{B}$ receives the tuple $(X_1, X_2, \hat{Y})$ (along with public parameters of $\mathbb{G}$ and the generator g) and should find the pair $(\hat{Z}_1, \hat{Z}_2)$ such that $\text{2dh}(X_1, X_2, \hat{Y}) = (\hat{Z}_1, \hat{Z}_2)$; $\mathcal{B}$ has access to the decision oracle $\text{2dhp}(X_1, X_2, ., ., .)$. $\mathcal{B}$ creates $\text{PK} = (X_1, X_2, g)$. $\mathcal{B}$ generates a challenge ciphertext as follows: it generates two random group members $(R_1, R_2 \leftarrow \mathbb{G})$ and sets the challenge ciphertext $c^* = (R_1, R_2, \hat{Y})$, this represents an encryption of a random string. $\mathcal{B}$ simulates $\mathcal{A}(\text{PK}, c^*)$. Whenever the simulated $\mathcal{A}$ asks a query $(c = (c_1, c_2, c_3), m = (m_1, m_2))$, $\mathcal{B}$ calls its own oracle with the query $\text{2dhp}(X_1, X_2, c_3, c_1/m_1, c_2/m_2)$ and forwards the response to $\mathcal{A}$. The simulation of $\mathcal{A}$ outputs $m = (m_1, m_2)$. $\mathcal{B}$ outputs $(\hat{Z}_1, \hat{Z}_2) = (R_1/m_1, R_2/m_2)$.

It is easy to check that the public-key generated in the simulation comes from the same distribution as legitimate keys. Since the challenge ciphertext in the OW-PCA experiment is chosen at random from the message space, which corresponds to a random pair of group elements, inspection of the OW-PCA scheme' encryption algorithm shows that the first two elements in the challenge ciphertext are random group elements, due to the fact that the message essentially acts as a one-time pad. Thus it is a quick check to see that again, the challenge ciphertext in the simulation comes from the same distribution as in the OW-PCA experiment. Finally, the decryption validity oracle outputs $\top$ on query $(c_1, c_2, R = g^y)$ if and only if $(X_1, X_2, c_3, c_1/m_1, c_2/m_2) = (g^{x_1}, g^{x_2}, g^y, g^{x_1 y}, g^{x_2 y})$, which is easy to check for each position of the tuples. □

Claim 15. *The encryption scheme presented in Figure 4.3 is key malleable. The algorithms (f, f', f_1, f_2, f_r) for which Property 1, 2 and 3 in Definition 4.3.1 holds is:*

Proof. For the encryption scheme $\mathbf{E}$ and the functions (f, f_1, f_2), we first prove in Claim 16 that Property 1 holds, then in Claim 17 we show that Property 2 holds, and then in Claim 18 we show that Property 3 holds.

— $f(\text{PK}_1 = (X_1, X_2, g))$:

1. $x'_1, x'_2 \leftarrow \{0, ..., |\mathbb{G}| - 1\}$
2. $X'_i \leftarrow X_i \cdot g^{x'_i}$ for $i \in \{1, 2\}$
3. $\text{PK}_2 \leftarrow (X'_1, X'_2, g)$
4. $s = (x'_1, x'_2)$
5. Output (PK_2, s)

— $f'(\text{SK} = (x_1, x_2), s = (x'_1, x'_2))$:

1. Output $(x_1 + x'_1, x_2 + x'_2)$

— $f_r(\text{PK} = (X_1, X_2, g), c = (c_1, c_2, c_3))$:

1. $r' \leftarrow \{0, 1, ..., |\mathbb{G}| - 1\}$
2. Output $(c_1 \cdot X_1^{r'}, c_2 \cdot X_2^{r'}, c_3 \cdot g^{r'})$

— $f_1(s = (x'_1, x'_2), c = (c_1, c_2, c_3))$:

1. $c'_i \leftarrow c_i \cdot c_3^{x'_i}$ for $i \in \{1, 2\}$
2. Output (c'_1, c'_2, c_3)

— $f_2(s = (x'_1, x'_2), c = (c_1, c_2, c_3))$:

1. $c'_i \leftarrow c_i \cdot c_3^{-x'_i}$ for $i \in \{1, 2\}$
2. Output (c'_1, c'_2, c_3)

Figure 4.4: THE ALGORITHMS (f, f', f_1, f_2, f_r) For **E**

Claim 16 (Alternate Indistinguishable Keys)**.** *For a fixed group $\mathbb{G}$ with the generator g for which the computational Diffie-Hellman assumption holds, we have that:*

$$\{(PK_1, SK_1) \leftarrow \textbf{\textit{Gen}}(1^k), (PK_2, SK_2) \leftarrow \textbf{\textit{Gen}}(1^k) : (PK_1, PK_2, SK_1, sk_2)\} \equiv$$
$$\{(PK_1, SK_1) \leftarrow \textbf{\textit{Gen}}(1^k), (PK_2, s) \leftarrow f(PK_1), sk_2 \leftarrow f'(SK, s) : (PK_1, PK_2, SK_1, SK_2)\}$$

*where **Gen** is the algorithm defined in Figure 4.3, and f_1 is defined in Claim 15.*

Proof. Notice that if $\text{SK}_1 = (x_1, x_2)$ then $\text{SK}_2 = (x_1 + x'_1, x_2 + x'_2)$ for randomly generated x'_1, x'_2. Hence the distribution of $(\text{PK}_1, \text{SK}_1, \text{PK}_2, \text{SK}_2)$ is identical to that of a pair of freshly generated public and secret keys.

Since these two distributions are identical, we conclude that with overwhelming probability $(\text{PK}_2, \text{SK}_2)$ are valid keys and hence the part (b) of the **Alternate Indistinguishable Keys** property

in Definition 4.3.1 holds.

□

Claim 17 (Inter-Key Ciphertext Malleability)**.** *For a fixed group $\mathbb{G}$ with the generator g for which the computational Diffie-Hellman assumption holds:*

For all k, for all $m \in \mathcal{M}$:

$$\Pr\Big[(PK, SK) \leftarrow \textbf{\textit{Gen}}(1^k), (PK', s) \leftarrow f(pk) : (\textbf{\textit{Enc}}_{PK}(m) \equiv f_1(s, \textbf{\textit{Enc}}_{PK'}(m))) \wedge (\textbf{\textit{Enc}}_{PK'}(m) \equiv f_2(s, \textbf{\textit{Enc}}_{PK}(m)))\Big] = 1.$$

where ***Gen*** *and* ***Enc*** *are the algorithms defined in Figure 4.3, and f_1 and f_2 are defined in Claim 15.*

Proof. Given that the keys generated in the Alternate Indistinguishable Keys property (Claim 16) come from the same distribution as legitimate keys, inspection of f_1 and f_2 shows that there is a one-to-one mapping between ciphertexts in the ranges of the two public-keys. □

Claim 18 (Ciphertext ReRandomization)**.** *For a fixed group $\mathbb{G}$ with the generator g for which the computational Diffie-Hellman assumption holds, we have that given $(PK, SK) \leftarrow g(1^k)$ and a ciphertext $c = \textbf{\textit{Enc}}_{PK}(m)$, then over the distribution of $S = (PK, m, c)$ we have that $\{f_r(PK, c)\}_S \approx_s \{\textbf{\textit{Enc}}_{PK}(m)\}_S$ where f_r is the algorithm defined in Figure 4.3.*

Proof. Notice that if c is encrypted under the random coin $r \leftarrow \{0, 1, ..., |\mathbb{G}| - 1\}$, then $f_r(\text{PK}, c)$ is an encryption with random coin $r + (r' \leftarrow \{0, 1, ..., |\mathbb{G}| - 1\})$ which is statistically close to a fresh encryption. □

□

4.6 Encryption Based on the Factoring Assumption

In this section, we provide an encryption scheme that is OW-PCA secure, and has the alternate-key generation and the inter-ciphertext malleability properties assuming the factoring assumption is intractable. The suggested encryption is similar to the encryption scheme secure against the chosen plaintext attack offered by Wee in [31] that is based on the factoring assumption. The only difference between the two is an additional consistency check in the decryption procedure.

Fix the Blum integer $N = PQ$ for safe primes $P, Q = 3 \pmod 4$ where $P = 2p + 1$ and $Q = 2q + 1$ for primes p, q. As shown in [53], the cyclic group of signed quadratic residues given by the quotient group $\mathbb{QR}_N^+ := \mathbb{QR}_N / \pm 1$ is of order pq and is efficiently recognizable given the modulus N. Assuming that factoring Blum integers are hard on average and that safe primes are dense, then the map $x \mapsto x^2$ (indexed by N) on the group $\mathbb{QR}_N^+$ is one-way. In other words, for all ppt adversaries $\mathcal{A}$, $\Pr[\mathsf{F}(\mathcal{A}, k) = 1]$ is negligible in the security parameter k, where the F experiment is defined as follows:

$\mathsf{F}(\mathcal{A}, k)$
1: Sample a Blum integer N with the security parameter k (i.e. $\lfloor \log p \rfloor = \lfloor \log q \rfloor = k$)
2: $x \leftarrow \mathbb{QR}_N^+$
3: $x' \leftarrow \mathcal{A}(N, x^2)$
4: Return $x' \stackrel{?}{=} x$

Claim 19. *The encryption scheme* $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *presented in Figure 4.5 is* OW-PCA *secure if the factoring assumption holds.*

Proof. To prove that **E** is OW-PCA secure, we need to show that for any ppt adversary $\mathcal{A}$ and security parameter $k \in \mathbb{N}$,

$$\Pr[\mathsf{OW\text{-}PCA}(\mathbf{E}, \mathcal{A}, k) = 1] \leq \epsilon(k)$$

We show this by a hybrid argument. Consider the following experiments:

Let $\mathbf{E} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ be any encryption scheme that is OW-PCA secure and Indistinguishable Key Generation, and inter-key ciphertext malleable with the message space $\{\mathcal{M}_k\}_{k\in\mathbb{N}}$, where $|\mathcal{M}_k| \in \omega(\log k)$, and a corresponding hard-core predicate $h(\cdot)$. Let $\Sigma = (\mathsf{GenKey}, \mathsf{Sign}, \mathsf{Verify})$ be a strong one-time signature scheme.

Gen(1^k)
1: SK $\leftarrow [(N-1)/4]$
2: PK $\leftarrow g^{2\text{SK}}$
3: Output (PK, SK)

Enc(PK, m)
1: $r \leftarrow [(N-1)/4]$
2: Output $(g^{2r}, (\text{PK}\cdot g)^r, g^r \cdot m)$

Dec$(\text{SK}, C = (c_1, c_2, c_3))$
1: **if** $c_1, c_2, c_3 \notin \mathbb{QR}_N^+$ **then** Output $\perp$
2: $R \leftarrow c_2 \cdot c_1^{-\text{SK}}$
3: **if** $R^2 \neq c_1$ **then** Output $\perp$
4: Output $c_3/R)$

Figure 4.5: THE ENCRYPTION SCHEME $\mathbf{E} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$

Experiment $\mathsf{OW\text{-}PCA}^{(1)}(\Pi, \mathcal{A}, k)$ modifies OW-PCA. It changes how decryption works in response to the adversary's queries. Fix the public and secret key $(\text{PK}, \text{SK}) \leftarrow \mathbf{Gen}(1^k)$. In $\mathsf{OW\text{-}PCA}^{(1)}(\Pi, \mathcal{A}, k)$, we answer the query $(c = (c_1, c_2, c_3), m)$ submitted by the adversary using the following subroutine (instead of returning $m \stackrel{?}{=} \mathbf{Dec}_{\text{SK}}(c)$):

$\mathbf{Verify}_{\mathbf{SK}}(c = (c_1, c_2, c_3), m)$

1. **if** $c_1, c_2, c_3 \notin \mathbb{QR}_N^+$ **then** Output $\perp$
2. $R \leftarrow c_3/m$
3. **if** $R^2 \neq c_1$ **then** Output $\perp$
4. **if** $c_2/R \neq c_1^{\text{SK}}$ **then** Output $\perp$ **o.w** Output $\top$

Claim 20.

$$\{\mathsf{OW\text{-}PCA}(\boldsymbol{E}, \mathcal{A}, k)\}_{k\in\mathbb{N}} \equiv \{\mathsf{OW\text{-}PCA}^{(1)}(\boldsymbol{E}, \mathcal{A}, k)\}_{k\in\mathbb{N}}$$

Proof. Follows by inspection that one oracle outputs $\top$ if and only if the other does. □

Experiment $\mathsf{OW\text{-}PCA}^{(2)}(\Pi, \mathcal{A}, k)$ modifies $\mathsf{OW\text{-}PCA}^{(1)}$ in two ways. First, in the key generation step, we compute the public key as $g^{2\mathrm{SK}-1}$ instead of $g^{2\mathrm{SK}}$. Second, we replace the checking condition in the forth step in the algorithm **Verify** by $c_2 \neq c_1^{\mathrm{SK}}$.

Claim 21.

$$\{\mathsf{OW\text{-}PCA}^{(1)}(\boldsymbol{E}, \mathcal{A}, k)\}_{k\in\mathbb{N}} \approx_c \{\mathsf{OW\text{-}PCA}^{(2)}(\boldsymbol{E}, \mathcal{A}, k)\}_{k\in\mathbb{N}}$$

Proof. First, we consider the indistinguishability of the public-keys in the two experiments. First imagine that the SK's were drawn from $\{0, ..., \phi(N)/4 - 1 = pq - 1\}$. Remember that the order of the group $\mathbb{QR}_N^+$ is pq. It is simple, therefore, to see that $g^{2\mathrm{SK}}$ and $g^{2\mathrm{SK}+1}$ both define uniform distributions over all possible elements in $\mathbb{QR}_N^+$. The statistical distance between the uniform distribution over $\{0, ..., \phi(N)/4 - 1\}$ and over $\{0, ..., (N-1)/4\}$ is negligible in $k \approx \log p \approx \log q$, and therefore the distributions in public-keys is indistinguishable.

To observe that modifying the **Verify** algorithm does not modify the experiment, note that when we have modified the public-key, valid encryptions of m transform from

$$(c_1 = g^{2r}, c_2 = (pk \cdot g)^r = g^{2\mathrm{SK}\cdot r} + r, c_3 = g^r \cdot m)$$

to

$$(c_1 = g^{2r}, c_2 = (pk \cdot g)^r = g^{2\mathrm{SK}\cdot r - r + r} = g^{2\mathrm{SK}}, c_3 = g^r \cdot m).$$

Therefore, with the modified PK in a valid encryption it is now expected that $c_1^{\mathrm{SK}} = c_2$. □

Claim 22. *For all $k \in \mathbb{N}$ and ppt adversaries $\mathcal{A}$, there exists a ppt adversary $\mathcal{B}$ such that:*

$$\{\mathsf{OW\text{-}PCA}^{(2)}(\boldsymbol{E}, \mathcal{A}, k)\}_{k\in\mathbb{N}} \equiv \{\mathsf{F}(\mathcal{B}, k)\}_{k\in\mathbb{N}}$$

Proof. We build the ppt adversary $\mathcal{B}$ that interacts with the F experiment in the outside while simulating the $\mathsf{OW\text{-}PCA}^{(2)}$ experiment for the adversary $\mathcal{A}$ in the inside. $\mathcal{B}$ receives N and x^2 from

the outside. $\mathcal{B}$ chooses a random generator g and SK $\leftarrow [(N-1)/4]$. $\mathcal{B}$ then sets PK $\leftarrow g^{2\text{SK}-1}$. $\mathcal{B}$ sets the challenge ciphertext $c^* = (x^2, (x^2)^{\text{SK}}, y)$ where y is a random element. Notice that the ciphertext c^* is an encryption of a random and unknown element m. $\mathcal{B}$ runs $\mathcal{A}$ on the input (PK, c^*). Whenever $\mathcal{A}$ asks a plaintext checking query, $\mathcal{B}$ answers as in the $\mathsf{OW\text{-}PCA}^{(2)}$ experiment. Finally $\mathcal{A}$ returns m' and halts. $\mathcal{B}$ returns to the outside y/m' and halts. Notice that if $m = m'$ then y/m' equals x. □

□

Claim 23. *The encryption scheme* $\boldsymbol{E} = (\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec})$ *presented in Figure 4.5 satisfies Definition 4.3.1.*

Proof. We first specify the algorithms (f, f', f_1, f_2, f_r) required in Definition 4.3.1:

— $f(\text{PK}_1)$:

1. Pick $x \in [N - 1/4]$
2. Compute $\text{PK}_2 \leftarrow \text{PK}_1 \cdot g^{2x}$
3. Output $(\text{PK}_2, s = x)$

—$f'(\textbf{SK}, s)$**:** Output $(\text{SK} + s)$

— $f_r(\text{PK}, c = (c_1, c_2, c_3))$

1. $r' \leftarrow [(N-1)/4]$
2. Output $(g^{2r'} \cdot c_1, (\text{PK} \cdot g)^{r'} \cdot c_2, g^{r'} \cdot c_3)$

— $f_1(s = x, c = (c_1, c_2, c_3))$:

1. $c_2' \leftarrow c_2 \cdot c_1^{-2x}$
2. Output (c_1, c_2', c_3)

— $f_2(s = x, c = (c_1, c_2, c_3))$:

1. $c_2' \leftarrow c_2 \cdot c_1^{2x}$
2. Output (c_1, c_2', c_3)

Figure 4.6: THE ALGORITHMS (f, f', f_1, f_2, f_r) For **E**

We now show that **E** satisfies the Alternate Key, Inter-key Malleable and Rerandomization properties.

Alternate Indistinguishable Keys For a randomly generated group with generator g, we have:

$$\{(\text{PK}_1, \text{SK}_1) \leftarrow \textbf{Gen}(1^k), (\text{PK}_2, s) \leftarrow f(\text{PK}_1), sk_2 \leftarrow f'(\text{SK}, s) : (\text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2)\}$$
$$\equiv \{\ldots(\text{PK}_1, \text{PK}_1 \cdot g^{2x}, \text{SK}_1, \text{SK}_1 + x)\}$$
$$\equiv \{\ldots(\text{PK}_1, g^{2(\text{SK}_1+x)}, \text{SK}_1, \text{SK}_1 + x)\}$$

Since x is chosen randomly from $[(N-1)/4]$, and since the distribution $\{2(\text{SK}_1 + x) \bmod \phi(N)/4\}$ in which both SK_1, x are uniform over $[(N-1)/4]$ is statistically close to the distribution $\{2(\text{SK}_1) \bmod \phi(N)/4\}$, it follows that the distribution above is statistically close to

$$\{(\text{PK}_1, \text{SK}_1) \leftarrow \textbf{Gen}(1^k), (\text{PK}_2, \text{SK}_2) \leftarrow \textbf{Gen}(1^k) : (\text{PK}_1, \text{PK}_2, \text{SK}_1, sk_2)\}$$

which establishes the first property.

Since these two distributions are statistically close, we conclude that with overwhelming probability in both mentioned distributions $(\text{PK}_2, \text{SK}_2)$ are valid keys and hence the part (b) of the **Alternate Indistinguishable Keys** property in Definition 4.3.1 holds.

Inter-Key Ciphertext Malleability For a randomly generated group $\mathbb{G}$ with the generator g, we argued above that with very high probability, the elements PK_2 and $g^{2(\text{SK}_1+x)}$ are statistically close. When this is true, then: (r is a random variable in this equation)

$$f_1(s, (c_1, c_2, c_3)) = (c_1 = g^{2r}, g^{(2(\text{SK}+x)+1)r} \cdot (g^{2r})^{-x}, g^r \cdot m)$$
$$= (c_1, g^{(2\text{SK}+1)r}, g^r \cdot m)$$
$$\equiv_s \textbf{Enc}_{\text{PK}_1}(m)$$

The statistical closeness in the last line follows because the random choice of x will cause some keys to be chosen with negligibly smaller probablity than others due to the ”wrap-around” inducing

difference between $(N-1)/4$ and pq. The other direction involving f_2 follows the same idea.

ReRandomization Let $c = (c_1, c_2, c_3) = (g^{2r}, (\text{PK} \cdot g)^r, g^r \cdot m)$. Then, applying $f_r(\text{PK}, c)$ we get $(g^{2(r+r')}, (\text{PK} \cdot g)^{r+r'}, g^{r+r'} \cdot m)$. Thus it suffices to show that the distribution on $r + r' \mod \phi(N)$ is statistically close to a $r \mod \phi(N)$ for $r \leftarrow [(N-1)/4]$ as defined in the **Enc** algorithm. Remember that $\mathbb{QR}_N^+$ is a cyclic group of order $\phi(N)/4 = pq$. Therefore, all that is needed to show is that for all fixed $r \in [pq]$ that the uniform distribution over $[(N-1)/4 = ((2p+1)(2q+1)-1)/4 = pq+p/2+q/2] \mod pq$ is statistically close to the uniform distribution over $[(N-1)/4] + r \mod pq$ for any fixed $r \in \phi(N)/4$. Thus in each experiment we have a near uniform distribution over pq, where $p/2 + q/2$ elements occur with probability $8/(N-1)$ as opposed to $4/(N-1)$. Therefore, in the worst case, for all but at most $p+q$ of the values in $\{1, ..., pq\}$ the probability of the value in each experiment is $4/(N-1)$ and so the difference in probabilities is 0, for the remaining $p+q$ the difference is at most $4/(N-1)$. Thus the statistical difference is less than $2(p+q)/pq \approx 1/2^k$.

□

Chapter 5

Multiparty Computation with Low Communication Overhead

5.1 Introduction

Secure computation with an honest majority can be accomplished without any cryptographic assumptions, but the best such protocol requires the parties to communicate $|f| \log |f| + d^2 \cdot poly(n, \log |f|)$ bits [54] and at least d rounds. Here $|f|$ is the size of the function being computed and d is the circuit depth of f, and thus the communication of the protocol is super-linearly related to the number of gates in f. Until recently, even the use of cryptographic assumptions for secure computation required $polylog(\lambda)$ communication overhead per gate [54] where λ is a security parameter.

Gentry[55] circumvents this per-gate overhead using fully homomorphic encryption schemes (FHE) as follows: the honest-but-curious parities use secure multi-party computation to generate an FHE key. Tthen each party encrypts its input and sends the resulting ciphertext and proof to other parties. Once all parties have encryptions of everyone's inputs, they compute the function of interest locally using the evaluation procedure of the FHE. Finally, they use the resulting ciphertexts as inputs to a secure multi-party computation to computes the decryption of the majority input. In order to be secure against malicious adversaries, the Naor and Nissim compiler [56], which makes heavy use of the PCP theorem, can be applied. The use of the PCP theorem in the SMC steps makes

the approach impractical, even when presented with a practical FHE scheme.

The motivation behind this work is to remove any use of the PCP theorem (or any other non-blackbox method such as generic ZK or NIZK) from the above framework for constructing communication-efficient secure protocols. In other words, we seek a black-box transformation from TFHE (Threshold Fully Homomorphic Encryption) to secure computation.

First Contribution The main technical hurdle in devising a black-box transformation from TFHE to secure computation is to implement the requirement for each player to prove that they "know the plaintext" corresponding to the encrypted input that they have broadcast. This step is essential because it prevents one player from copying (or mauling via the homomorphism) the input of a player who has acted earlier. To handle this step, we show how to construct a two-round black-box proof of knowledge of an encrypted bit for any circuit private FHE scheme using only the encryption scheme. Since our protocol is only two rounds, it is not zero-knowledge (cf. [57]) but can provably keep the encrypted bit hidden. Our POK requires that the public-key contain a labeled encryption of 0 and 1, which given all known FHE schemes seems to be a natural modification. [1] For traditional FHE schemes, the POK can be used completely black-box, without even the need for the modification.

The basic idea of our proof of knowledge protocol is to first modify the encryption scheme so that the message is encoded using an error-correcting code (ECC) based verifiable secret sharing (VSS) scheme. To encrypt a message we first generate its secret shares, and encrypt them independently using fresh randomness. A verifier now requests the Prover to reveal the randomness used to encrypt a sub-threshold number of the shares. The verifier then does a consistency check, based on the ECC underlying the scheme, to ensure that the shares were encoded properly. In particular, the error-correcting code we choose offers a property that allows one to check whether local parts of the codeword are error-free. The verifier accepts if everything appears to be properly coded. Since the number of shares revealed is less than the threshold, it does not leak any information about the

[1] Since all current schemes contain bit-wise encryptions of their own secret-keys which are random bit strings, and a natural extension of any protocol that provides encryptions of one's own secret-key can be used to derive a labeled encryption of 0 and 1 which we describe.

original message. To show a proof of knowledge property, we argue that an extractor can rewind the Prover and ask for another set of shares to be opened. With high probability, this second transcript provides enough new shares to run the VSS recover algorithm and recover the original message. The one issue with this approach is that the Prover must reveal the randomness used to encrypt some of the shares. The semantic security of an encryption scheme does not guarantee any security when these random bits are revealed—in particular, the security of the rest of the unopened encryptions are not guaranteed. Instead, we require the encryption scheme to be secure against a selective opening attack (SOA). Fortunately, a result of Hemenway et al. [58] can be generalized to show that any circuit private homomorphic encryption scheme can be made into an SOA-secure one.

We point out that our proof of knowledge requires the encryption scheme to be homomorphic and circuit-private. Recently, Damgård et al. [59] demonstrates a three-round Sigma-protocol for knowledge of plaintext, but their protocol *requires* the underlying encryption scheme to also be homomorphic on the random coins used to encrypt. Although many FHE schemes support this property on their random coins, it is certainly not specified in the definition of FHE. In contrast, circuit privacy has been independently defined and seems to be a naturally weaker property.[2] Moreover, their scheme requires the message space for the FHE to be over $\mathbb{Z}_N$ for N related to the security parameter. While in general, single-bit FHE implies many-bit FHE, we are not aware of any such transformation that *also* preserves the homomorphism over the random coins as required by their protocol. Thus, the requirement for large message space *and* homomorphism over the random coins seem to be extra assumption which our work can avoid (our protocol also works on single-bit FHE). Finally, the Sigma protocol from [59] must eventually be compiled into a full zero-knowledge protocol using standard techniques; these techniques add round complexity and/or setup assumptions whereas we show that our two-round protocol with its hidden-bit property suffices for our secure computation protocol.

[2]Even though current schemes achieve circuit privacy via randomness homomorphisms, it is certainly plausible for future constructions to achieve circuit privacy in other ways. Moreover, there do not seem to be any natural ways to transform a circuit private scheme to one with a randomness homomorphism, and thus we feel it is a weaker notion.

Second Contribution By combining our result above with almost any TFHE scheme, we then present a secure multi-party protocol that avoids *both* per-gate communication complexity *and* PCP theorems or any non-blackbox use of a cryptographic primitive. The communication complexity of our protocol is $O(\lambda^c \cdot n^2)$ where λ is a security parameter and c is a small constant for the TFHE scheme and is thus independent of $|f|$. Our black-box transformation is particularly important because if practical FHE (and TFHE) can be constructed, our transformations will result in practical SFE, whereas Gentry's scheme relying on PCP proofs and SMC will have significant overhead. Our work is in the standard model and does not require extra trust assumptions such as the common reference string, random oracle or the public key model.

Final Contribution For completeness, we also construct a *threshold* fully homomorphic public key encryption scheme (TFHE) based on the Approximate GCD problem and the fully homomorphic encryption scheme presented by van Dijk et al. [60], and our result was the first to demonstrate the feasibility of directly achieving this threshold primitive for FHE.

We present our protocols in the information-theoretic model over secure point-to-point channels, and thus our protocols are secure in the presence of an honest majority. Thus, when used with our transformation, the resulting protocol is also only secure with an honest majority. By using another TFHE that tolerates a dishonest majority, our transformation results in an secure computation protocol that also tolerates the same.

The TFHE scheme provides a constant-round protocol for n players to generate a public key and distribute private shares of the corresponding secret key of a fully homomorphic encryption scheme. This step itself is non-trivial since the generation of the public key for an FHE scheme (that is based on bootstrapping) requires encryption of the secret key. Later, a majority of players can cooperatively decrypt a ciphertext by running a constant-round protocol on their private shares and a public ciphertext. We also provide methods for distributed encryption and for proving knowledge of an encrypted value.

We note that both our TFHE key generation and decryption protocols are more efficient than generically applying secure function evaluation techniques to the key generation or decryption

algorithms of an FHE scheme. For example, with the right set of the parameters, our decryption protocol requires only a constant number of share multiplications, whereas generic techniques would require $O(poly(\lambda))$ such multiplications. We heavily exploit the linear nature of the operations involved in key generation, encryption and decryption for the particular FHE scheme of van Dijk et al. For key generation and decryption, we develop specific multiparty computation protocols that evaluates an arithmetic circuit using verifiable secret sharing techniques, that would be more efficient than the application of generic techniques.

Comparison With Other FHE-based Secure Computation Protocols Gentry's [55] secure computation protocol was the first to achieve communication complexity that is independent of $|f|$ by using the PCP theorem in several steps.

Asharov, Jain and Wichs [61] and López-Alt, Tromer, and Vaikuntanathan [62] have constructed more efficient TFHE schemes based on LWE and the closely related RLWE assumption, which can be reduced to varying degrees to worst-case lattice problems. Their approaches rely on the ability to construct an FHE that also has a homomorphism on the secret keys, and can also be used to achieve secure computation with communication that is independent of $|f|$. Together, our results demonstrate that the TFHE primitive can be developed from reductions to different classes of hardness assumptions, and therefore TFHE is not simply a consequence of a specific hardness properties.

To achieve security against malicious adversaries, López-Alt et al. rely on a common reference string setup so that players can use a NIZK to prove to each other that their keys and their input ciphertexts are well-formed. The use of such NIZK also requires additional hardness assumptions, since (T)FHE is not known to imply NIZK. They can also instantiate their ideas in the standard model by replacing these NIZK proofs with traditional interactive ZK proofs; but in either case, the generic (NI)ZK techniques used will require non-blackbox use of the underlying TFHE scheme.[3] By choosing the CRS model, the authors observed that by using a more expensive simulation-sound

[3]In other words, the encryption algorithm of the TFHE will need to be expressed in terms of a graph-coloring instance (or Hamiltonicity or circuit-sat etc). As far as we know, this transformation requires a high-order polynomial overhead.

NIZK, their protocols can also achieve UC-security. Our protocols only claim standard security, but as a reviewer has noted, it is likely that we can state some of ours results as UC in a TFHE-hybrid model. We are actively pursuing this direction.

Asharov et al. use efficient Sigma Protocol constructions to prove well-formedness; these make heavy use of the underlying mathematical structure of the LWE assumption. In order to have efficient NIZK proofs, they must rely on the use of the Random Oracle model and the Fiat-Shamir heuristic to transform the Σ-protocols into NIZK proofs. In any case, due to the black-box nature of our SMC construction, with simple modifications to the public-key to include labeled ciphertexts representing encryptions of 0 and 1, either of the López-Alt et al. or Asharov et al. TFHE schemes can be plugged in to our construction to achieve security against an arbitrary number of malicious adversaries, with abort. In contrast, with our scheme we are guaranteed output delivery, but need an honest majority of players.

The protocols of Damgård et al. [59] and Bendlin et al. [63] use a different approach to constructing secure computation protocols from traditional homomorphic encryption. Their schemes rely on the idea from Beaver [64] for circuit randomization. First, they use an offline phase in which the parties use a somewhat homomorphic encryption primitive to create shares of triples (a, b, c) such that $a \cdot b = c$. One triple is required for each multiplication gate in f that is to be evaluated and requires approximately $O(n/s)$ "heavy" cryptographic operations to generate. Next, after such triples have been created, the parties use only information-theoretic methods to evaluate the circuit. This approach results in admirable communication parameters for small circuits (as they have also run practical examples); nonetheless, the approach requires linear communication for each gate in $|f|$, and thus does not achieve our main aim of eliminating this relationship.

Finally, these prior results are all in a model in which n parties are computing, and the protocols can tolerate up to $n - 1$ malicious parties. In contrast, our protocols require an honest majority. The relative incomparability of these models is well understood. In particular, in the model that tolerates up to $n - 1$ malicious adversaries, if any one party deviates form the protocol or fails, then all parties output $\perp$. Alternately, with an honest majority, all parties can output an effective output,

as supported by our protocol. For a discussion of the relative merits of the two models, and the impossibility of having protocols that achieve the best of both worlds for general functionalities, see the work of Ishai et al. [65].

In summary, all of these recent works have advantages and disadvantages of their own; our major contribution is the black-box transformation and the independent hardness assumption.

Related work The notion of threshold cryptography scheme was implicitly motivated by Shamir in [66] and was formally introduced by Desmedt et al. [67]. Several extensions and schemes have been considered in the literature for different public encryption and signature schemes. Cramer, Damgård and Nielson [68] along with Jakobbsson and Juels [69] show how to use threshold cryptography to construct secure multiparty computation protocols. In more detail, we use many ideas from [68] which shows how a homomorphic threshold cryptosystem can be used to achieve general multiparty computation protocols. We leave as an open question whether there exists a black-box technique for generating a threshold FHE scheme from any FHE scheme.

5.2 Preliminaries and Notation

Basic Notations Let a be a string split into n shares held by n players, and let an adversary control a set of players $I \subset [n]$ We denote the shares of a held by adversary the adversary i as $[^{\star}a]^I$.

Definition 5.2.1. *(Threshold Fully Homomorphic Encryption Scheme) A 4-tuple of protocols and algorithms* $(\boldsymbol{Gen}, \boldsymbol{Enc}, \boldsymbol{Dec}, \boldsymbol{Eval})$ *is a threshold fully homomorphic encryption scheme if the following hold:*

Key Generation *An n-party protocol **Gen** that at each invocation returns a new public key PK and the secret key $(SK_1, \ldots, SK_n)$, where SK_i is the share of the secret key for Player$_i$.*

Encryption *A ppt algorithm $\boldsymbol{Enc}_{PK}(m, r)$ that returns the encryption of the plaintext m under the public key PK with random coins r.*

Decryption *There exists a ppt n-party protocol* $\textbf{\textit{Dec}}(c, SK_1, \ldots, SK_n)$*, which returns the plaintext m using the shares SK_i held by honest party Player$_i$, where $c = \textbf{\textit{Enc}}(m, r)$ for some random r.*

f_{PK}-homomorphic *There exists a ppt algorithm* ***Eval*** *which given a polynomial f, ciphertexts $c_1 \in \textbf{\textit{Enc}}_{PK}(m_1), \ldots, c_k \in \textbf{\textit{Enc}}_{PK}(m_k)$ for some k and a public key PK outputs $c \in \textbf{\textit{Enc}}(f(m_1, \ldots, m_k))$.*

Threshold Indistinguishability The natural notion of chosen plaintext attack indistinguishability security needs to be modified in the venue of threshold cryptography to take into account the fact that the adversary has access to shares of the secret-key, and we need to ensure these do not aid it. We now give the definition of security for a threshold encryption scheme.

Definition 5.2.2 (Threshold Indistinguishability [68])**.** *Let $\mathcal{A}$ be an efficient adversary that on input 1^k, a public-key PK and any set C of the corresponding secret-key shares $SK_1, ..., SK_n$, where $(PK, SK_1, \ldots, SK_n)$ are generated by the execution of the key generation protocol, outputs two messages m_0 and m_1 and state information s. Hence $(m_0, m_1, s) \leftarrow \mathcal{A}(1^k, PK, C, \{SK_c\}_{c \in C})$. Let $(s, c_i) \leftarrow X_i(k, C)$ denote the distribution over (s, c_i) where $c_i \leftarrow \textbf{\textit{Enc}}(PK, m_i)$. Then $X_i = \{X_i(k, C)\}_{k \in \mathbb{N}, C}$ for $i \in \{0, 1\}$ are ensembles, and we require that $X_0 \approx_c X_1$ when $|C| < n/2$.*

Standard security notions for secure multi-party computation protocols can be used to define the security for the protocols **Gen** and **Dec** in any given instantiation of a TFHE (e.g., we can consider security in the real/ideal standalone paradigm, the UC framework, etc..)

Circular Threshold Security Notice that the security definition for a threshold encryption scheme is not defined for the case where the adversary has access to the encryption of the secret key. We present a definition to capture the notion of security for this case, which we call circular threshold security for a semantically secure encryption scheme. Our definition is based on the circular security definition in [70], but we must take in to account the fact that the adversary has access to shares of the secret-key, $[^\star\text{SK}]^I$. The definition needs to ensure that these shares do not affect the circular security.

Definition 5.2.3 (Circular Threshold Security)**.** *Let* $\Pi = (\textbf{\textit{Gen}}, \textbf{\textit{Enc}}, \textbf{\textit{Dec}})$ *to be a semantically secure threshold encryption scheme. For algorithm* $\mathcal{A}$ *and set* $C \subset \{1, \ldots, n\}$ *and random* $k \in \mathbb{N}$*, let* $\mathsf{IND\text{-}CirThrCPA}_b(\Pi, \mathcal{A}, C, k)$ *to be the output of the experiment described in Figure 5.1.*

$\mathsf{IND\text{-}CirThrCPA}_b(\Pi, \mathcal{A}, C, k)$

1: $(\text{PK}, \text{SK}_1, \ldots, \text{SK}_n) \leftarrow \textbf{Gen}(k)$. Let SK be the secret key for PK.
2: $m_0 \leftarrow \text{SK}$
3: $m_1 \leftarrow 0^{|\text{SK}|}$
4: $c^* \leftarrow \textbf{Enc}(\text{PK}, m_b)$
5: Output $\mathcal{A}([^\star\text{SK}]^C, \text{PK}, c^*)$

Figure 5.1: THE CIRCULAR THRESHOLD SECURITY DEFINITION

Encryption scheme Π is circular threshold secure if for all ppt algorithms $\mathcal{A}$ and any set C where $|C| < n/2$, it holds that the following two ensembles are computationally indistinguishable:

$$\{\mathsf{IND\text{-}CirThrCPA}_0(\Pi, \mathcal{A}, C, k)\}_k \approx_c \{\mathsf{IND\text{-}CirThrCPA}_1(\Pi, \mathcal{A}, C, k)\}_k$$

Selective Opening Security We present the definition for selective opening security. Intuitively, a scheme that is secure against selective openings ensures that the release of randomness used to encrypt some ciphertexts cannot be used to compromise the security of other encryptions with the same public-key.

Definition 5.2.4 (IND-SO-SEC Encryption Security)**.** *A public-key encryption scheme* $\Pi = (G, E, D)$ *is Indistinguishable Selective Opening secure if, for any message sampler* M *that supports efficient conditional resampling and any ppt adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and all sufficiently large* k*:*

$$\mathsf{Ind\text{-}SO\text{-}Real}(\Pi, \mathcal{A}, M, k) \approx_s \mathsf{Ind\text{-}SO\text{-}Ideal}(\Pi, \mathcal{A}, M, k)$$

A message sampler M *is a ppt algorithm that outputs a vector* $\vec{m}$ *of* n *messages from a given distribution. It is an efficient conditional resampler if, when given two auxiliary inputs a set of indices* $I \subseteq [n]$ *and a vector of messages* $\vec{m} = (m_1, \ldots, m_n)$*,* M *will sample another vector* $\vec{m'} =$

$(m'_1, \cdots, m'_n)$ *conditioned on* $m_i = m'_i$ *for each* $i \in I$. *We define the experiments* Ind-SO-Real *and* Ind-SO-Ideal *in Figure 5.2.*

Ind-SO-Real$(\Pi, \mathcal{A}, M, k)$
1: $(\text{PK}, \text{SK}) \leftarrow G(k)$
2: $\vec{m} = (m_1, \ldots, m_n) \leftarrow M$
3: Generate random coins $r_1, \ldots, r_n$
4: $(I, \sigma) \leftarrow \mathcal{A}_1(\text{PK}, E_{\text{PK}}(m_1, r_1), \ldots, E_{\text{PK}}(m_n, r_n))$
5: $b = \mathcal{A}_2(\sigma, (m_i, r_i)_{i \in I}, \vec{m})$

Ind-SO-Ideal$(\Pi, \mathcal{A}, M, k)$
1: $(\text{PK}, \text{SK}) \leftarrow G(k)$
2: $\vec{m} = (m_1, \ldots, m_n) \leftarrow M$
3: Generate random coins $r_1, \ldots, r_n$
4: $(I, \sigma) \leftarrow \mathcal{A}_1(\text{PK}, E_{\text{PK}}(m_1, r_1), \ldots, E_{\text{PK}}(m_n, r_n))$
5: $\vec{m'} = (m'_1, \ldots, m'_n) \leftarrow M_{I, \vec{m}[I]}$
6: $b = \mathcal{A}_2(\sigma, (m_i, r_i)_{i \in I}, \vec{m'})$

Figure 5.2: REAL VS. IDEAL MODEL FOR SELECTIVE OPENING SECURITY

Bootstrapping a Ciphertext We introduce the notion of bootstrapping a ciphertext. Gentry introduced the notion of Bootstrapping to reduce noise in a somewhat fully homomorphic encryption scheme, in order to achieve a fully homomorphic scheme. In contrast, we assume the existence of an FHE and simply use it to reduce noise produced in ciphertexts generated in our selective opening attack secure scheme.

Definition 5.2.5. *(Bootstrapping a Ciphertext) For a FHE scheme* $\Pi = (G, E, D, \textbf{\textit{Eval}})$ *and the security parameter* k, *let* D_Π *be* Π*'s decryption circuit, which takes a secret key and s ciphertext as input. Given a ciphertext* C *encrypted with respect to a public-key PK and secret-key* $SK = (SK_1, .., SK_\ell)$ *we require that PK contains a bit-wise encryption of SK, denoted* $s_1, ..., s_\ell$ *where* $s_i = E(PK, SK_i)$. *Let* $(C_1, .., C_n)$ *denote the bits of* C, *and generate* $c_i = E(PK, C_i)$. *We say that the value* $C^\dagger = \textbf{\textit{Eval}}(PK, D_\pi, s_1, ..., s_\ell, c_1, .., c_n)$ *(which homomorphically evaluates* $D(SK, C)$*) is the result of bootstrapping* C.

Lossy Encryption We give the definition of a lossy encryption scheme below, which is a natural modification of the notion of lossy functions. We can generate traditional keys which encrypt

and decrypt properly, or we can generate lossy keys in which the distributions of encryptions over all messages are statistically indistinguishable (and thus correct decryption can clearly not be satisfied on average). Finally, we require that the distributions of traditional keys and lossy keys are computationally indistinguishable.

Definition 5.2.6 (Lossy Encryption)**.** *A lossy public-key encryption scheme is a triple* (G, E, D) *of ppt algorithms such that:*

Correctness of Injective Keys: *For all* $(PK, SK) \leftarrow G(k, \mathsf{INJ})$*,* $b \in \{0,1\}$*, and random strings* r*, it holds that* $D(SK, E(PK, b, r)) = b$*.*

Lossiness of Lossy Keys: *For all* $(PK, SK) \leftarrow G(k, \mathsf{LOSSY})$*:* $E(PK, 0) \approx_s E(PK, 1)$*.*

Computational Indistinguishability of Lossy and Injective Keys: *Define* $s : (x, y) \mapsto x$ *to project pairs of public- and secret-keys to pulblic-keys.*

$$\{s(G(1^k, \mathsf{INJ}))\}_k \approx_c \{s(G(1^k, \mathsf{LOSSY}))\}_k$$

Openability: *The following is implied by lossiness. There exists an algorithm (not necessarily poly-time)* Opener *which when given a lossy public-key PK,* $(PK, SK) \leftarrow G(k, \mathsf{LOSSY})$ *and a ciphertext* $c \leftarrow E(PK, b)$*, will with non-negligible probability (over the choice of* PK, SK *and random bits used to generate* c*) output two strings* r_0 *and* r_1*, such that* $E(PK, 0, r_0) = c$ *and* $E(PK, 1, r_1) = c$*.*

Circuit Privacy for a Homomorphic Encryption We give the definition for Circuit Privacy below. Intuitively, a homomorphic encryption is circuit private if the distribution of evaluating a circuit on a ciphertext is close to that of fresh ciphertexts.

Definition 5.2.7. *([Statistical] Circuit Private Homomorphic Encryption). A homomorphic encryption scheme* ε *is circuit-private for circuits in a set* C_ε *if, for any key pair* (PK, SK) *output by* $\boldsymbol{Gen}(\lambda)$*, any circuit* $C \in C_\varepsilon$*, and any fixed ciphertext* $\psi = \langle \psi_1, \ldots, \psi_t \rangle$ *that are in the image of*

Enc$_\varepsilon$ *for plaintexts* $\pi_1, \ldots, \pi_t$*, the following distributions (over the random coins in* ***Enc***$_\varepsilon$*,* ***Eval***$_\varepsilon$*) are [statistically] indistinguishable:*

$$\textbf{\textit{Enc}}_\varepsilon(PK, C(\pi_1, \ldots, \pi_t)) \approx \textbf{\textit{Eval}}_\varepsilon(PK, C, \psi)$$

In the original schemes first presented by both Dijk et al. [60] and Gentry [55], the initial evaluation functions are deterministic and not circuit-private. In order to overcome this problem, both works introduce a method for adding random noise to encryptions, whether they are output from **Eval** or **Enc**, and thus in some sense rerandomizing them. This is done by adding an 'encryption' of 0 to the ciphertext in question, but where the 'encryption' has significantly more noise than would be generated by either the legitimate encryption or evaluation process. Specifically, they introduce ppt algorithms labeled $\mathsf{CircuitPrivacy} : \mathcal{C}_b \to \mathcal{C}'_b$, where $\mathcal{C}_b$ consists of all the ciphertexts that are output from $\textbf{Enc}_{\text{PK}}(b)$ or a call to **Eval** with an encrypted output bit of b. It is the case that for any b and any $c_{b,0}$, $c_{b,1} \in \mathcal{C}_b$ where $c_{b,0}$ is a ciphertext that is output from $\textbf{Enc}_{\text{PK}}(b)$ and $c_{b,1}$ is a ciphertext that is output from call to **Eval** with an encrypted output bit of b :

$$\mathsf{CircuitPrivacy}(c_{b,0}) \approx_s \mathsf{CircuitPrivacy}(c_{b,1}).$$

In the case of the construction based on approximate GCD, $\mathsf{CircuitPrivacy}(c)$ chooses a random subset $S \subseteq \{1, \ldots, \tau\}$, and $r \leftarrow [-2^{\eta-6}, 2^{\eta-6}]$ and output $c' \leftarrow [c + \sum_{i \in S} x_i]_{x_0} + 2r$ (See Appendix C of [60] for more details). Gentry describes a similar method for achieving circuit privacy on lattice based encryptions [55].

Verifiable Secret-Sharing Scheme A $\binom{n}{n/2+2}$ Verifiable Secret-Sharing scheme consists of a sharing algorithm which takes as input a secret s and produces n-shares $s_1, ..., s_n$. These shares have the property that for any $T \subset \{1, \ldots n\},\ |T| < n/2 + 2$ it is the case that $\{s_i\}_{i \in T}$ is information theoretically independent from s. However, for any $S \subseteq \{1, \ldots n\}$ s.t. $|S| \geq n/2 + 2$, it is the case that the reveal algorithm, when given $\{s_i\}_{i \in S}$, can reconstruct s. In a traditional interactive setting we require that all non-cheating parties agree on the reconstructed secret.

We use a modification of the Cramer et al. [71] verifiable secret sharing scheme. We note that in our application, we do not need to deal with interactive adversaries, nor players, so the scheme is significantly simplified. We present the sharing and revealing algorithms below. It is assumed that all of the operations are in some finite-field F of appropriate size.

Protocol 1 . $VSShare_{\binom{n}{n/2+2}}(s)$

1: Choose a random degree $n/2+1$ bi-variate polynomial f such that $f(0,0) = s$
2: Share $s_i = (\vec{a}, \vec{b}) = (i, (f(i,1), \ldots, f(i,n)), (f(1,i), \ldots, f(n,i)))$
3: $r_1, \ldots, r_n \leftarrow R$
4: Output $s_1, ..., s_n$

Protocol 2 . $VSReveal_{\binom{n}{n/2+2}}(s_1, ..., s_{n/2+2})$

1: For each $s_i = (i, \vec{a}_i, \vec{b}_i)$ ensure that a_i and b_i are $n/2+2$-consistent
2: If not output $\perp$
3: For each $i \neq j$ ensure s_j, s_i are pairwise-consistent
4: If not output $\perp$
5: Interpolate f, based on shares
6: Output $f(0,0)$

Figure 5.3: THE PROTOCOLS $VSShare_{\binom{n}{n/2+2}}$ AND $VSReveal_{\binom{n}{n/2+2}}$

Definition 5.2.8. *A vector* $(e_1, ..., e_n) \in F_n$ *is* $n/2+2-$consistent *if there exists a polynomial* w *of degree at most* $n/2+1$ *such that* $w(i) = e_i$ *for* $0 \leq i < n$.

Definition 5.2.9. *Given two shares* $s_i = (i, \vec{a}_i = (a_{i1}, \ldots, a_{in}), \vec{b}_i = (b_{1i}, \ldots, b_{ni}))$ *and* $s_j = (j, \vec{a}_j(a_{j1}, \ldots, a_{jn}), \vec{b}_j = (b_{1j}, \ldots, b_{nj}))$, *we say that they are* pairwise consistent *if* $a_{ij} = b_{ij}$ *and* $a_{ji} = b_{ji}$.

Definition 5.2.10. *For our purposes it is useful to note that given the* $n \times n$ *matrix*

$$\begin{bmatrix} f(1,1) & f(1,2) & \ldots & f(1,n) \\ f(2,1) & f(2,2) & \ldots & f(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ f(n,1) & f(n,2) & \ldots & f(n,n) \end{bmatrix},$$

that a share s_i simply corresponds to the i^{th} row and column of the matrix. We will call this the matrix representation of the shares. Notice that when given in the matrix representation, any two shares are necessarily pairwise consistent. Given a set of n pairwise consistent shares $\vec{s} = (s_1, ..., s_n)$, we define $M_{\vec{s}}$ as the $n \times n$ matrix representation of the shares.

5.3 Proof of Knowledge of an Encryption

As noted in the Introduction, the method of Cramer, Damgård, and Nielsen [68] requires an honest-verifier zero-knowledge proof of knowledge of encrypted values for the threshold schemes that they employ. We provide a weaker 2-round solution to that requirement, which alas, is not zero-knowledge but also does not release any information about the bit being discussed (we formalize this below). Moreover, our construction only makes black-box use of the underlying circuit-private FHE scheme.

We construct this proof through a two-step process. At a high-level, instead of encrypting a bit b, we will use a specific $\binom{n}{n/2+2}$ verifiable secret sharing scheme to generate n shares of b and encrypt those shares. [4] In order to give a proof of knowledge of the encryption of b, we will allow a verifier to select $n/2 + 1$ of the encryptions of shares of b, and then direct the Prover to decommit them by revealing the randomness used to encrypt them. To extract the bit, our extractor rewinds the proof and selects an alternate $n/2 + 1$ shares, so that with high probability, it can use $n/2 + 2$ shares to reconstruct b, and only b due to the verifiability of the secret sharing scheme. The problem with this approach is that revealing the randomness for an encryption raises selective decommitment issues. We use techniques from Hemenway et al. [58] to construct a bit-wise Indistinguishable Selective-Opening Secure encryption scheme from our threshold fully-homomorphic scheme. We can then use it to bitwise encrypt the VSS shares.

We note that the encryptions of the shares under the bit-wise Indistinguishable Selective-Opening Secure scheme is not itself a homomorphic encryption scheme. For example, we cannot multiply directly two sets of shares encoding b_0 and b_1 and expect the result to encode $b_0 \cdot b_1$. However,

[4] We use a verifiable secret sharing scheme with a $n/2 + 2$ threshold to simplify the proof of the VSS, thus $|T| = n/2 + 1$ is chosen to be right under the threshold of the VSS, as one might expect.

the individual encrypted bits are still properly encoded ciphertexts under the FHE scheme, but having had a circuit-privacy evaluation function applied to them. Intuitively, therefore, we can homomorphically evaluate the reveal function of the secret sharing scheme to get a single encryption representing the reconstituted bit. This encryption can then be used to homomorphically evaluate the function as in Cramer et al. [68]. There is however a snag: in principle, once the circuit-privacy function has been applied to a ciphertext, it may no longer be able to have homomorphic operations applied to it, as this is not guaranteed by the definition.[5] However, this problem is easily surmounted by applying Gentry's bootstrapping technique (cf. Definition 5.2.5) to re-encode the selective-opening secure schemes in to ciphertexts which can have homomorphic operations applied to them, and thus the VSS's reveal algorithm can be applied to the individual bits of the shares, resulting in ciphertext of the encoded bit, which is in the ciphertext space of the TFHE scheme.

5.3.1 Using FHE to construct a Selective Opening Encryption Scheme

Hemenway et al. [58] show how any re-randomizable encryption scheme can be used to construct a natural lossy encryption scheme and thus, by the result of Bellare et al. [72], is secure against indistinguishable selective opening attacks (IND-SO-SEC; see Definition 5.2.4).

Thus lossy encryption provides the ability to generate keys that produce "lossy" encryptions in which the distribution of encryptions of 0 is statistically close to the distribution of encryptions of 1. These keys are indistinguishable from regular injective keys. An alternate notion of security for encryption relates to the selective opening problem (Definition 5.2.4). In this somewhat counterintuitive notion, an adversary is given a sequence of ciphertexts corresponding to different messages and can query for the random coins used to encrypt a subset of those ciphertexts. Upon receiving these coins, the adversary then attempts to break the secrecy of the other ciphertexts. In other words, a scheme that is secure against selective openings ensures that the release of randomness used to encrypt some ciphertexts cannot be used to compromise the security of other encryptions with the same public-key. We note that CPA security is implied by the definition of

[5] Further, in practice, with known schemes, these ciphertexts have too much noise in them to allow further homomorphic operations without sacrificing decryption correctness.

Lossy encryption, see [58] for details.

Since the Hemenway and Ostrovsky construction relies on re-randomization, they suggest that the distribution of a "fresh" encryption of a message should be statistically close to a rerandomization of a fixed message. They point out that all homomorphic encryption schemes up to that point achieved this property by adding an encryption of 0 to the current message. While this property was true of all schemes at the time, it is not actually true of the known fully homomorphic encryption schemes, because each time we add an encrypted message to another we increase the amount of noise that is embedded in the ciphertexts, and thus fresh encryptions have less noise than encryptions that have had operations (such as addition) applied to them. Fortunately, the property they state is overly strong, and a simple observation shows that for their construction to go through they only require that the distributions

$$\{r \leftarrow R : E(pk, 0, r) \boxplus E(pk, m, r_0)\} \approx_s \{r \leftarrow R : E(pk, 0, r) \boxplus E(pk, m, r_1)\},$$

for all public-keys PK, messages m and random strings r_0 and r_1 where $\boxplus$ is the homomorphic addition operation. However, it is simple to see that even these two distributions are not statistically close for the fully homomorphic encryption schemes that have been proposed. Fortunately, both schemes under consideration have rerandomization functions built to ensure *Circuit-Privacy*, as is defined in Definition 5.2.7.

5.3.2 Construction of a SOA from Lossy

The essential idea is to perform normal (non-lossy) encryption, we generate a public-key for the Lossy scheme by generating a traditional public-key and secret-key for the TFHE, and then we augment the public-key with two labeled ciphertexts c_0 and c_1, representing encryptions of 0 and 1. Now, to actually encrypt a bit b, we take c_b, and rerandomize it using the circuit-privacy function (In comparison, Hemenway and Ostrovsky add an encryption of the bit 0). Decryption works as it does in the FHE scheme. The lossy key generator simply has c_1 represent an encryption of 0 instead of 1. By the CPA security of the TFHE scheme, the keys are indistinguishable. The scheme is formally

described below.

Key Generation $G'(k, b), b \in \{\mathsf{INJ}, \mathsf{LOSSY}\}$**:** Let $(\text{PK}, \text{SK}) \leftarrow G(k), c_0 \leftarrow E(\text{PK}, 0)$, $c_1 \leftarrow E(\text{PK}, 1)$ and $c_1' =\leftarrow E(\text{PK}, 0)$. If $b = \mathsf{INJ}$ Output $\text{PK}' = (pk, c_0, c_1)$ and $\text{SK}' = \text{SK}$, else when $b = \mathsf{LOSSY}$ output $\text{PK}' = (\text{PK}, c_0, c_1')$ and $\text{SK}' = \text{SK}$.

Encryption $E'(\mathbf{PK}' = (\mathbf{PK}, c_0, c_1), b)$**:** Output $\mathsf{ReRand}(c_b)$.

Decryption $D'(\mathbf{SK}, c)$**:** Output $D(\text{SK}, c)$.

Theorem 5.3.1. *If (G, E, D) is a circuit-private FHE, then the blackbox construction (G', E', D') described in Section 5.3.2 is an IND-SO-SEC secure encryption scheme.*

Proof. Follows from [58] and [72]. □

5.3.3 Modifying the SOA-secure Encryption Scheme to Support POKs

Again, in order to be able to provide a proof of knowledge that the a party has knowledge of the value encrypted, we need to provide a POK. We will show a 2-round public-coin proof of knowledge of the encrypted bit based on any selective opening secure scheme. The protocol is neither zero-knowledge nor witness indistinguishable but does maintain secrecy of the encrypted bit. First, we encrypt bits using the following protocol. Let $\Pi' = (G', E', D')$ be the selective-opening attack secure scheme described in Theorem 5.3.1. We construct a new encryption scheme $\hat{\Pi} = (\hat{G}, \hat{E}, \hat{D})$ to encode bits properly so we can give proofs of knowledge about them that keep the encrypted bit hidden. We define $\hat{G} = G'$, and give the algorithms for $\hat{E}$ and $\hat{D}$ in Figure 5.4.

5.3.4 Hidden Bit POK

Given a ciphertext $\mathbf{C} = \{c_{i,j}\}_{1 \leq i,j \leq n}$ output by our encryption algorithm $\hat{E}$ and the random strings used to generate it, $\vec{r}$, we show how to perform a two-round proof of knowledge of the encrypted bit $\hat{D}(\text{SK}, \mathbf{C})$ in Figure 5.5. P will prove that it has knowledge of the underlying shares of the verifiable secret-sharing scheme that have been encrypted, and thus the bit that has been encrypted. In order to do this, the verifier sends a random challenge of indices $T \subset \{1, \ldots, n\}$, where $|T| = n/2 + 1$.

$\hat{E}(\text{PK}, b, r)$

1: $(s_1, ..., s_n) \leftarrow VSShare_{\binom{n}{n/2+2}}(b)$
2: Let M be the $n \times n$ matrix representation of shares $(s_1, \ldots, s_n)$
3: $c_{i,j} = E'(\text{PK}, M_{i,j}, r_{i,j})$ (These are bitwise encryptions of M)
4: Output $\mathbf{C} = \{c_{i,j}\}_{1 \le i,j \le n}$

$\hat{D}(\text{SK}, \mathbf{C})$

1: $M = \{M_{i,j}\}_{1 \le i,j \le n} \leftarrow D'(\text{SK}, \mathbf{C})$
2: Let $(s_1, \ldots, s_n)$ be the shares corresponding to matrix M
3: $T' = \{t | 1 \le t \le n$ share s_t is $n/2 + 2$ -consistent$\}$
4: If $|T'| < n/2 + 2$ output $\perp$
5: Let $T \subseteq T'$ s.t. $|T| = n/2 + 2$
6: Output $VSReveal_{\binom{n}{n/2+2}}(s_{t_1}, ..., s_{t_{n/2+2}})_{t_i \in T}$

Figure 5.4: THE MODIFIED SOA-SECURE ENCRYPTION SCHEME TO SUPPORT POKS

The encryptor then decommits to these encryptions by providing the random-bits used to encrypt each share of the bit. If each bit decommits successfully, and the result is $n/2 + 1$ valid shares to the VSS, then the verifier accepts.

Prover(PK, $\mathbf{C} = \{c_{i,j}\}_{1 \le i,j \le n}$ $= \hat{E}(\text{PK}, b, r), M, r)$		Verifier(PK,$\mathbf{C} = \{\mathbf{c_{i,j}}\}_{\mathbf{1 \le i,j \le n}}$)
Let $c_{i,j} = E'(\text{PK}, M_{i,j}, r_{i,j})$	$\xleftarrow{T}$	$T \leftarrow \{S \mid S \subset \{1, ..., n\} \wedge \lvert S \rvert = n/2 + 1\}$
	$\xrightarrow{\{M_{i,x}, r_{i,x}, M_{x,i}, r_{x,i}\}_{\substack{i \in T \\ 1 \le x \le n}}}$	if $\exists i, j$ s.t. $c_{ij} \neq E'(\text{PK}, M_{i,j}, r_{i,j})$, output $\perp$.
		Output 1.

Figure 5.5: A TWO-ROUND PROOF OF KNOWLEDGE OF THE ENCRYPTED BIT $\hat{D}(\text{SK}, \mathbf{C})$

Completeness

Follows by inspection.

Extractability (Soundness)

Soundness follows from an extractor.

$Extractor(\mathbf{C}, \text{PK}, U_1 = \{M_{i,x}, r_{i,x}, M_{x,i}, r_{x,i}\}_{\substack{i\in T_1 \\ 1\le x\le n}}, U_2 = \{M_{i,x}, r_{i,x}, M_{x,i}, r_{x,i}\}_{\substack{i\in T_2 \\ 1\le x\le n}})$

1: Let $T = T_1 \cup T_2,\ U = U_1 \cup U_2$

2: If $|T| < n/2$ output $\perp$

3: If $\exists i \in T, x \in \{1, \ldots, n\}$ s.t. $E'(\text{PK}, M_{i,x}, r_{i,x}) \neq c_{i,x}$ or $E'(\text{PK}, M_{x,i}, r_{x,i}) \neq c_{x,i}$ output $\perp$

4: For each $i \in T$ reconstruct its corresponding share s_i

5: Output $VSReveal_{\binom{n}{n/2+2}}(s_{r_1}, ..., s_{r_{n/2}})$, where $r_1, \ldots, r_{n/2}$ are the smallest n/2 indices in T

Figure 5.6: THE *Extractor* TO PROVE THE SOUNDNESS FOR THE POK PROTOCOL

Theorem 5.3.2. *For all sufficiently large n, for all $d > 0$, for all $(SK, PK) \leftarrow \hat{G}$, for all 'ciphertext' inputs C, and provers P', if $(P', V)(C = \{c_{i,j}\}_{1\le i,j\le n}, PK)$ accepts with probability $1/n^d$, then there exists a probabilistic polynomial time extractor that, with all but negligible probability, outputs a set of decommitments to all ciphertexts for a given set of indices $L = \{\ell_1, \cdots, \ell_{n/2+2}\} \subseteq [n]$ that constitute shares $S = \{s_{\ell_1}, ..., s_{\ell_{n/2+2}}\}$ such that $VSReveal_{\binom{n}{n/2+2}}(s_{\ell_1}, ..., s_{\ell_{n/2+2}}) = \hat{D}(SK, C)$.*

Definition 5.3.3. *We say an $n \times n$ matrix representation of shares has $t-$ consistent indices, if there is a set S of size t such that for each $i \in S$, each row i and column i is $n/2 + 2$ consistent.*

Proof. Given the ability to rewind the prover-verifier protocol, we can extract the encrypted bit by recovering enough shares of the VSS scheme. We continue to execute the prover/verifier protocol until we get two distinct separate accepting proofs. It is a simple observation that except with exponentially small probability, we will succeed in $O(n^{d+1})$ rewinds. Let (T_1, U_1) and (T_2, U_2) be the flows in the first and second accepting proofs, respectively. By the security of the commitment scheme (Here we are using our encryption scheme as a simple commitment scheme), the probability that there is a ciphertext $c_{i,j}$ that is ever decommitted to in two distinct fashions is negligble.

We feed these inputs in to *Extractor* in Figure 5.6. If there is not a valid encryption of a bit (fewer than $n/2 + 2$ committed and consistent shares), then by Lemma 5.3.4, the probability that the verifier outputs anything other than $\perp$ is less than $\frac{1}{\binom{n}{n/2+2}}$ which grows exponentially small.

Given the decommitments of the shares $\{s_i\}_{i_\in T_i}$ for different randomly chosen set of indices T_1 and T_2, note these sets are not the same by selection, and therefore there is no chance that $\perp$ is

output by the extractor. Next the extractor executes a $VSReveal_{\binom{n}{n/2+2}}$ command. However, this is not necessarily over the same shares as would be revealed in a legitimate decryption. We need to ensure that no matter which of the rewound and newly played legitimate traces we receive, we are going to reveal the same encrypted bit, with all but negligible probability. That is, we need to ensure that $VSReveal_{\binom{n}{n/2+2}}(s_{r_1}, ..., s_{r_{n/2}}) = VSReveal_{\binom{n}{n/2+2}}(s_1, ..., s_{n/2})$. This is the case, as shown in Lemma 5.3.5 because of the verifiable properties of the secret sharing scheme ensures that even in the case of a corrupted dealer (improper ciphertext encoding of shares) then all honest players will reveal the same value, with all but negligible probability. Therefore, with all but negligible probability we have that the extractor outputs the same value as $\hat{D}(\mathbf{SK}, \vec{c})$. □

Lemma 5.3.4. *Let M be an $n \times n$ matrix with at most $n/2+1$ consistent indices. The probability that any $n/2+1$ randomly selected indices (without replacement) choose a set of $n/2+1$ consistent indices is no more than*

$$1/\binom{n}{n/2+1}.$$

Proof. There can be at most 1 set of size $(n/2+1)$ that is $(n/2+1)$ consistent in an $n \times n$ matrix. The lemma follows by computing the probability of choosing this one set from a set of n objects. □

Lemma 5.3.5. *Let M be $n \times n$ matrix representation of shares. Let $S, T \subseteq \{1, \ldots, n\}$, $|S| = |T| = n/2+2$, $S \neq T$, and the rows $R_S = \{r_i\}_{i \in S}$, $R_T = \{r_i\}_{i \in T}$ and columns $C_S = \{c_i\}_{i \in S}$, $C_T = \{c_i\}_{i \in T}$ are all $n/2+2-$consistent. Let $s = (s_1, ..., s_{n/2+2})$ and $t = (t_1, ..., t_{n/2+2})$ be the shares drawn from M corresponding to the sets of indices S and T respectively. Then $VSReveal_{\binom{n}{n/2+2}}(s_1, ..., s_{n/2+1}) = VSReveal_{\binom{n}{n/2+2}}(t_1, ..., t_{n/2+1})$.*

Proof. Note that in $VSReveal_{\binom{n}{n/2+2}}$ lines 1–4 will never output $\perp$ under our conditions, so all that we need do is show that f will interpolate to the same value in both cases.

We know that the rows $R_T = \{r_i\}_{i \in T}$ and columns $C_R = \{c_i\}_{i \in T}$ are all $(n/2+2)-$consistent. Choose any $j \in S \setminus T$. Let $T = \{t_1, \ldots, t_{n/2+2}\}$. Consider $c_j = (c_{1,j}, c_{2,j}, \ldots, c_{n,j})^T$. Since c_j is $n/2+2-$consistent, the points $(c_{t_1,j}, t_1), \ldots, (c_{t_{n/2+1},j}, t_{n/2+2})$, interpolate to a unique univariate

degree $n/2+1$ polynomial (i.e. $f(x,j)$). This defines $(c_{1,j}, c_{2,j}, \ldots, c_{n,j})^T$, so the column j must be consistent with T. Since the jth column was an arbitrary column in S different from those in T, all such columns must be consistent with the rows defined be T. A symmetric argument shows that rows selected by S must be consistent with the columns selected by T. Therefore, both sets are consistent in that they define the same polynomials. Therefore, interpolation in $VSReveal_{\binom{n}{n/2+2}}$ will result in the same output. □

Hidden Bit

We show that no efficient cheating verifier can predict the bit b, when given $C = \hat{E}(\text{PK}, b, r)$ as a theorem for which we are engaging in a POK.

Theorem 5.3.6. *For every ppt adversary $A = (A_1, A_2)$, there exists a negligible function μ such that $\Pr[HB_A(1^k) = 1] \leq 1/2 + \mu(k)$, where the experiment HB is defined in Figure 5.7.*

$HB_A(1^k)$
1: $(\text{PK}, \text{SK}) \leftarrow \hat{G}(1^k)$
2: $b \in \{0,1\}$
3: $C = \{c_{i,j}\}_{1 \leq, i,j \leq n} = \hat{E}(\text{PK}, b)$ where $c_{i,j} = E'(\text{PK}, M_{i,j}, r_{i,j})$ are the SOA encryptions
4: $(T, \sigma) \leftarrow A_1(\text{PK}, C)$ where $T \subset \{1, \ldots, n\}$, $|T| = n/2+1$
5: $b' \leftarrow A_2(\sigma, (M_{i,j}, r_{i,j})_{i,j \in T})$
6: Output 1 iff $b = b'$

Figure 5.7: THE HB EXPERIMENT

Proof. This follows directly from the IND-SO-SEC security of $\Pi' = (G', E', D')$. Suppose an adversary $A = (A_1, A_2)$ breaks the hidden bit security of the protocol. That is for some $d > 0$ and infinitely many k: $\Pr[HB_A(1^k) = 1] \geq 1/2 + 1/k^d$. We use it to build an adversary $B = (B_1, B_2)$ and message selector M that breaks the IND-SO-SEC security (cf. Def. 5.2.4, page 85) of $\Pi' = (G', E', D')$. The message selector M chooses a random bit b, let $(s_1, ..., s_n) \leftarrow VSShare_{\binom{n}{n/2+2}}(b)$, and let $\mathbf{M}$ be the $n \times n$ matrix that represents the shares $(s_1, \ldots, s_n)$ according to the ECC representation of the VSS. Output $\mathbf{M}$.

The adversary $B_1\left(\text{PK}, (E(\text{PK}, \mathbf{M_{i,j}}, \mathbf{r_{i,j}}))_{1\leq i,j\leq n}\right)$ for the IND-SO-SEC experiment simulates $(T, \sigma) \leftarrow A_1(\text{PK}, C = (E(\text{PK}, \mathbf{M_{i,j}}, \mathbf{r_{i,j}}))$, and outputs $I = \{(i,j)|1 \leq i,j \leq n, i \in T \text{ or } j \in T\}$ and $\sigma' = (T, \sigma)$. Recall by the definition of A_1, $|T| = n/2 + 1$.

The conditional message selector $M_{I,\vec{m}[I]}$ from the SOA security definition finds a random bi-variate polynomial of degree $n/2+1$ in each variable over the field F such that $f(0,0) \in \{0,1\}$ and for each $(i,j) \in I$, it holds that $f(i,j) = \mathbf{M}_{i,j}$. Since $|T| = n/2+1$, and thus we have effectively release $n/2+1$ shares for a VSS scheme that requires $n/2+2$ for reconstruction, the information secrecy property of the VSS guarantees there are exactly the same number of such selections for the case $f(0,0) = 0$ and $f(0,0) = 1$. $M_{I,\vec{m}[I]}$ outputs $\{f(i,j)\}_{1\leq,i,j\leq n}$.

The adversary $B_2(\sigma, (M_{i,j}, r_{i,j})_{(i,j)\in I}, \mathbf{M}^*)$ computes the shares $(s_1^*, ..., s_n^*)$ that correspond to $\mathbf{M}^*$, and runs $VSReveal_{\binom{n}{n/2+2}}(s_1^*, .., s_n^*) = b'$, it then executes $b \leftarrow A_2(\sigma, (m_{i,j}, r_{i,j})_{(i,j)\in I})$ and outputs 1 iff $b = b'$.

Now consider $\Pr[B_\Pi^{\mathsf{Ind\text{-}SO\text{-}Real}}(1^k) = 1]$, this is a perfect simulation of $HB_A(1^k)$, and therefore by the assumption that A breaks the hidden-bit security is at least $1/2+\epsilon$, where $\epsilon \geq 1/k^c$. In contrast, consider $\Pr[B_\Pi^{\mathsf{Ind\text{-}SO\text{-}Ideal}}(1^k) = 1]$. In the case that $VSReveal_{\binom{n}{n/2+2}}(s_,^*1..., s_n^*) = VSReveal_{\binom{n}{n/2+2}}(s_1, ..., s_n)$, which occurs with probability exactly $1/2$, it is again a perfect simulation $HB_A(1^k)$, and so outputs 1 with probability $1/2+\epsilon$. In contrast, when $VSReveal_{\binom{n}{n/2+2}}(s_,^*1..., s_n^*)) \neq VSReveal_{\binom{n}{n/2+2}}(s_1, ..., s_n)$, then we know that A_2 outputs $VSReveal_{\binom{n}{n/2+2}}(s_1, ..., s_n)$ with probability $1/2+\epsilon$, and so B_2 must output 1 with probability $1-(1/2+\epsilon) = 1/2-\epsilon$. Therefore, $\Pr[B_\Pi^{\mathsf{Ind\text{-}SO\text{-}Ideal}}(1^k) = 1] = (1/2)(1/2+\epsilon+1/2-\epsilon) = 1/2$. Therefore, $\Pr[B_\Pi^{\mathsf{Ind\text{-}SO\text{-}Real}}(1^k) = 1] - \Pr[B_\Pi^{\mathsf{Ind\text{-}SO\text{-}Ideal}}(1^k) = 1] = 1/2+\epsilon-1/2 \geq 1/k^c$, breaking IND-SO-SEC security. □

Using the SOA Ciphertexts in a Secure Multiparty Computation Protocol In our SMC construction, we will encode all users' inputs using the POK scheme above. The encrypted inputs are sent to the other parties. After each party's input has been confirmed with a proof of knowledge, the parties homomorphically evaluate the different ciphertexts to get an appropriate encrypted output. However, as explained before, the POK encryptions are not themselves homomorphic. To solve this problem we use Gentry's bootstrapping technique. Bootstrapping lets us take a ciphertext in

an FHE scheme with any amount of noise that still allows for proper decryption (specially, this is potentially more noise than is permissible to perform any extra homomorphic operations without destroying the correctness of the ciphertext), and output a new ciphertext in the FHE scheme, of the same value, but with a small enough amount of noise that it can be properly computed on through the use of the FHE's evaluation function. Given a ciphertext $C = \{c_{i,j}\}_{1 \leq i,j \leq n}$ in the POK scheme, each $c_{i,j}$ is a ciphertext from a lossy encryption scheme. To convert C into a corresponding encryption $C^{\dagger}$ in the TFHE scheme we do the following: We bootstrap each $c_{i,j}$ which is simply a TFHE ciphertext that has had the circuit-privacy function applied to it—thus containing potentially too much noise to apply further homomorphic operations to, but not so much that it decrypts improperly— to receive the corresponding lower-noiseTFHE ciphertext $c'_{i,j}$. The c' ciphertexts can now be evaluated in the THFE eval function, and in particular we can use the TFHE eval function, to evaluate $VSReveal_{\binom{n}{n/2+2}}$. The result of this evaluation is the ciphertext $C^{\dagger}$ corresponding to the output.

Protocols vs. Algorithms We note that there is one technical issue that needs to be resolved, which is that in this section we have described the key generation and decryption algorithms as stand-alone algorithms, rather than protocols. For our purposes, we need a joint protocol for key generation and decryption. For this reason, we need to modify our key generation algorithm in the TFHE scheme to include an encryption of the bits 0 and 1 in the public-key. These values allow the parties to encrypt under the SOA secure encryption scheme $\hat{\Pi}$. The SOA secure scheme does not modify the decryption algorithm, so there is no need for modification to the decryption protocol.

5.4 Threshold FHE for the Integers

We briefly summarize the FHE scheme $\Pi = (G, E, D, \mathbf{Eval})$ based on the Approximate-GCD problem described by [60]. The scheme relies on the boot-strapping principle and is based on a somewhat homomorphic scheme parameterized by the following variables:

γ is the bit-length of the integers in all the somewhat homomorphic scheme's public key,

τ is the number of integers in the somewhat homomorphic scheme's public key,

η is the bit-length of the secret key in somewhat homomorphic scheme (which is the hidden approximate-gcd of the integers in the public-key),

ρ is the bit-length of the noise.

For the security parameter λ, vanDijk [60] suggests the following relationships:

- $\rho = \omega(\log \lambda)$, to protect against brute-force attacks on the noise
- $\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$, in order to support homomorphism for deep enough circuits to evaluate the "squashed decryption circuit"
- $\gamma = \omega(\eta^2 \log \lambda)$, to thwart various lattice-based attacks on the underlying approximate-gcd problem
- $\tau \geq \gamma + \omega(\log \lambda)$, in order to use the leftover hash lemma in the reduction to approximate gcd
- $\rho' = \rho + \omega(\log \lambda)$

Now the parameters are set as following: $\rho = \lambda$, $\rho' = 2\lambda$, $\eta = \tilde{O}(\lambda^2)$, $\gamma = \tilde{O}(\lambda^5)$ and $\tau = \gamma + \lambda$.

For $z \in \mathbb{R}$, define $r_p(z) = z - \lfloor z/p \rceil \cdot p$, i.e. it is the remainder in the range $(-p/2, p/2)$. For a specific (ρ-bit) odd positive integer p, let the distribution $D_{\gamma,\rho}(p)$ over γ bit integers be:

$$D_{\gamma,\rho}(p) = \{\text{choose } q \leftarrow Z \cap [0, 2^\gamma/p), r \leftarrow Z \cap (-2^\rho, 2^\rho) : \text{output } x = pq + r\}$$

5.4.1 Public Key Homomorphic Encryption Scheme

The somewhat homomorphic encryption scheme $\Pi' = (G', E', D', \textbf{Eval}')$ works as follows:

$G'(\lambda)$ The secret key is an odd η-bit integer: $p \leftarrow (2Z + 1) \cap [2^{\eta-1}, 2^\eta)$.

For the public key, sample $x_i \leftarrow D_{\gamma,\rho}(p)$ for $i = 0, \ldots, \tau$. Relabel the vector $\vec{x}$ so that x_0 is the largest integer. Restart unless x_0 is odd and $r_p(x_0)$ is even. The public key is $pk = \langle x_0, x_1, \ldots, x_\tau \rangle$.

$E'(pk, m \in \{0,1\})$ Choose a random subset $S \subseteq \{1, 2, ..., \tau\}$ and a random integer r in $(-2^{\rho'}, 2^{\rho'})$. Output $c^* \leftarrow [m + 2r + 2\sum_{i \in S} x_i]_{x_0}$.

Eval'$(pk, C_\varepsilon, c_1, ..., c_t)$ Given an (arithmetic) admissible circuit C_ε with t inputs, and t ciphertexts c_i, apply the (integer) addition and multiplication gates of C_ε to the ciphertexts, performing all the operations over the integers, and return the resulting integer.

If $f(x_1, \ldots, x_t)$ is the multivariate polynomial representation of the circuit C_ε, and f is of degree d, then we say that C_ε is admissible if:

$$d \leq \frac{\eta - 4 - \log |f|}{\rho' + 2}$$

where $|f|$ is the l_1 norm of the coefficient vector of f.

$D'(sk, c)$ Output $m' \leftarrow (c \mod p) \mod 2$.

The scheme can be transformed into a fully homomorphic one by applying the bootstrapping transformations described in [60]. We call the new evaluate procedure **Eval**. In particular, the decryption depth of the circuit must be squashed by adding extra information to the public key and modifying the encryption procedure. Towards this goal, set the additional parameters $\kappa = \gamma\eta/\rho'$, $\Theta = \omega(\kappa \log \lambda)$, and $\theta = \lambda$. Define the scheme $\Pi = (G, E, D, \textbf{Eval})$ as follows:

$G(\lambda)$ Generate $sk^* = p$ and pk^* as before. Set $x_p \leftarrow \lfloor 2^\kappa/p \rceil$. Choose at random a Θ-bit vector $\vec{s} = \langle s_1, \ldots, s_\Theta \rangle$ with Hamming weight θ, and let $S = \{i : s_i = 1\}$. Choose at random integers $u_i \in Z \cap [0, 2^{\kappa+1})$ for $i = 1, \ldots, \Theta$, subject to the condition that $\sum_{i \in S} u_i = x_p$ (mod $2^{\kappa+1}$). Set $y_i = u_i/2^\kappa$ (it is a real number with $\lceil \log \theta + 3 \rceil$ bits of precision) and $\vec{y} = \langle y_1, \ldots, y_\Theta \rangle$. Output the secret key SK $= (\vec{s})$ and public key PK $= (pk^*, \vec{y})$ (Notice that p is no longer needed in the secret key).

$E(m, \textbf{PK})$ Generate a ciphertext c^* as before in E' (i.e., an integer). Then for $i \in [\Theta]$, set $z_i \leftarrow [c^* \cdot y_i]_2$, keeping only $\lceil \log \theta \rceil + 3$ bits of precision after the binary point for each z_i. Output both c^* and $\vec{z} = \langle z_1, \ldots, z_\Theta \rangle$.

$D(c = \langle c^*, \vec{z} \rangle, \mathbf{SK})$ Output $m' \leftarrow [c^* - \lfloor \sum_i s_i z_i \rceil]_2$.

Theorem 5.4.1 (Implicit from [60]). *Fix the parameters $(\rho, \rho', \eta, \gamma, \tau)$ and (κ, Θ, θ) as noted above. Under the assumption that the (ρ, η, τ) approximate-GCD problem and the (θ, Θ)-sparse subset sum problem are hard, Π is a semantically-secure compact bootstrappable fully homomorphic encryption scheme.*

In the rest of this section we discuss the construction of a TFHE scheme $\tilde{\Pi} = (\tilde{G}, \tilde{E}, \tilde{D}, \tilde{\mathbf{Eval}})$ from this particular FHE scheme. We point out that in any such transformation $\tilde{E} = E$ and $\tilde{\mathbf{Eval}} = \mathbf{Eval}$, and thus we only need to describe protocols for computing $\tilde{G}$ and $\tilde{D}$.

5.4.2 Secret Sharing Primitives

Let F_q denote the finite field $\mathbb{Z}/q\mathbb{Z}$ with q being a prime. Let $[^\star a]$ denote a secret-sharing of $a \in F_q$ and $a \leftarrow \text{REVEAL}([^\star a])$ denote the execution of the reconstruction protocol for a in which the parties use their shares of $[^\star a]$ as input to reconstruct the shared secret a. We refer to the simulator for REVEAL protocol as $S_{\text{REVEAL}}([^\star a]^I, a')$, which takes as input the shares of a known by adversary (denoted $[^\star a]^I$), and reveal the value a'. This is done in a manner that is indistinguishable from the real world execution. Throughout the chapter, in algorithmic descriptions we do not specify the field that secrets are shared in. *The field order must be large enough to handle the integers encountered in the fully homomorphic computations without causing wrap-around* but is otherwise arbitrary. One can determine the field order needed by computing the maximum of the required orders for each FHE algorithm. Therefore in all secret sharing instances, we refer to the field order as an ℓ bit prime.

Let $[^\star a]_B$ denote a shared value which is bit-decomposed. That is, every player holds a share of each bit of a. Also $[^\star a]_{i..j}$ denotes the bit-by-bit share of the the substring of a from the index i to the index j.

We assume that the secret sharing scheme is linear. Hence parties that hold the shares $[^\star a]$ and $[^\star b]$ can compute shares for $[^\star a + b]$ and for $[^\star ac]$ for a public constant c without interacting. We also assume that there is an unconditionally secure protocol MULT to compute $[^\star ab]$ from the shares $[^\star a]$

and $[^\star b]$. Such a linear secret sharing scheme and a corresponding *constant round* multiplication protocol MULT that can be instantiated with protocols described in [66] and [73]. We refer to the simulator for this protocol as $S_{\text{MULT}}([^\star a]^I, [^\star b]^I, res)$ which takes as input the shares of a and b held by the adversary and reveal the value res as the result of executing the protocol in an indistinguishable way from the real world execution for the adversary. Since we assume the existence of the multiplication protocol as an abstract primitive, we express the round complexity as the number of sequential multiplication calls that are necessary during the protocol. We also express the communication complexity as the number of the total multiplication calls during the protocol.

Some Known Primitives We first describe some known primitives from previous works on secret sharing. The protocols are used as building blocks to construct our shared key-generation and decryption protocols for the threshold decryption scheme. These protocols are all secure against static dishonest minority and make use of atomic broadcast channels. We also describe slight modification to some of them to fit our applications.

$\text{RAN}_2()$ Damgård et al. [74] give an n-party protocol RAN_2 in which the players get no input and receive as output shares of a uniformly distributed random bit $a \in \{0, 1\}$. The protocol requires 2 sequential multiplication rounds. The simulator for this protocol is $S_{\text{RAN}_2}(b)$. At the end of running the simulator, each party holds a share of b.

$\text{COMP}([^\star a]_B, [^\star b]_B)$ [74], [75], and [76] introduced a method to compare the bit-wise values $[^\star a]_B$ and $[^\star b]_B$ in a multiparty setup.The returned value is 1 if $a \geq b$, and 0 otherwise. The protocol runs in 8 rounds (two of which are preprocessing rounds) and invokes $13\ell + 6\sqrt{\ell}$ multiplications where ℓ is the bit length of the order of field in which a and b are shared. By adding another $\ell + \sqrt{\ell}$ multiplications, as mentioned in [76], the comparison can be extended to return two bits that determine the case when $a = b$ as follows:

$$\text{COMP}([^\star a]_B, [^\star b]_B) = \begin{cases} ([1],[0]) & \text{if } [a]_B > [b]_B \\ ([0],[0]) & \text{if } [a]_B = [b]_B \\ ([0],[1]) & \text{if } [a]_B < [b]_B \end{cases}$$

BITS$([^\star a])$ This protocol provides a method for taking $[^\star a]$ and computing shares for each of the bits in the bit-wise representation of a. The protocol returns $\ell = \lceil \log a \rceil$ shares of bits $([^\star a_0], \ldots, [^\star a_{\ell-1}])$, where $a = \sum_i a_i 2^i$. The protocol takes 23 rounds (7 of which are preprocessing rounds) and invokes $31\ell \log \ell + 71\ell + 30\sqrt{\ell}$ multiplications (ℓ is the bit length of the field order a is shared in). The simulator for this protocol is denoted $S_{\text{BITS}}([^\star a]^I, a_{\ell-1}, \ldots, a_0)$. Each party would hold a share of values a_0, ..., $a_{\ell-1}$ at the end of the simulation. This algorithm was presented in [76].

SOLVED-BITS(k) This protocol returns shares of each bit of the bit-wise representation of a value chosen uniformly at random in the range $[0, k]$. The protocol takes seven rounds (two of which are preprocessing rounds) and $52\ell + 24\sqrt{\ell}$ multiplications. The simulator for this protocol is $S_{\text{SOLVEDBITS}}(a)$, where a is the simulated random output. This algorithm was presented in [76].

power$([^\star x], k)$ This algorithm returns shares of x^k in the case that x is an invertible field element. The simulator for this protocol is $S_{\text{POWER}}([^\star x]^I, k, y)$. The simulator should give the shares of y to represent the shares of x^k to all parties during the simulation. This protocol is a simple modification of a protocol based on the works of [77] and [74] which given as input shares $[^\star x_1], \ldots, [^\star x_k]$ returns in constant rounds shares for all of the values $1 \leq i < j \leq k$, $\prod_{k=i}^{j} x_k$. The protocols takes 5 rounds and incurs $5\ell + k - 1$ multiplications.

MULT$^*([^\star x_1], \ldots [^\star x_k])$ For the case all x_i's are invertible field members, [77] and [74] suggests a protocol to compute the $x_i \cdot x_{i+1} \cdot \ldots \cdot x_j$ for all $1 \leq i < j \leq k$ in constant rounds. The round complexity would be 5 with $5\ell + k - 1$ invocation of MULT. The simulator for this protocol would be $S^*_{\text{MULT}}([^\star x_1]^I, \ldots [^\star x_k]^I, x)$. At the end of simulation, each party should

hold a shares of x as the multiplication of $x_1, \ldots, x_k$. For the case all x_i are the same and we need x^i for $i \in [k]$, we write it as $\text{MULT}^*([^\star x], k)$ and we define the simulator accordingly.

$\text{MOD}([^\star x], m)$ This protocol computes shares of $x \mod m$, where m is a public value. The simulator for this protocol is $S_{\text{MOD}}([^\star x], m, a)$, where the simulator produces shares of a, with the intention that $a = x \mod m$. The protocol takes up to 40 rounds, and $31\ell^2 \log \ell + 31\ell \log \ell + 84\ell^2 + 71\ell + 36\ell\sqrt{\ell} + 30\sqrt{\ell}$ multiplications. This protocol is a natural augmentation of a protocol presented in [74] which results from using techniques in [76] to improve efficiency.

5.4.3 Sharing the Public and Secret Key

In this section, we describe a constant-round n-party protocol to generate both a public key and shares of the secret key for the fully homomorphic encryption scheme Π (For technical reasons, the public key will also contain encryptions of 0 and 1 for use in our POK). Our schemes rely on the secret sharing sub-protocols described in Section 5.4.2.

Recall that the secret-key for Π consists of a Θ-bit vector $\vec{s}$ with Hamming weight θ. Our first modification to Π is to note that instead of θ, it suffices to select a vector with Hamming weight in the interval $\theta \pm \theta/4$. This observation allows us to pick a SK by independently flipping coins that are 1 with probability θ/Θ.

Generating the public key is more complicated. The public key consists of the vectors $\vec{x}$ and $\vec{u}$. There are several steps in generating the public key. In order to generate $\vec{u}$, we first compute the shares of $\vec{s}$ and the shares of p which is an odd integer in the interval $[2^{\eta-1}, 2^\eta)$. Second, we compute $x_p = \lfloor 2^\kappa / p \rceil$. Using $\vec{s}$ and x_p, we compute the vector $\vec{u}$ using the formula $\vec{u} = \sum_i s_i \cdot u_i \mod 2^{\kappa+1}$. Third, using bits of $1/p$ computed in previous steps, we generate the x_i's. Forth, we generate encryptions of the secret key $\vec{s}$. In the next sections, we provide more details on each of these steps.

Producing the SK $\vec{s}$

The secret key for the squashed scheme consists of a random Θ-bit vector $\vec{s} = (s_1, \ldots, s_\Theta)$ with Hamming weight θ. We argue that setting the Hamming weight of $\vec{s}$ to be any value in the range $\theta \pm \theta/4$ does not affect the security or correctness of the scheme. To verify this, note that the sparse subset-sum problem is assumed to be hard for $\theta = \Theta^\epsilon$ for $0 < \epsilon < 1$; our change does not violate this condition. Also, our new range of settings for θ does not increase the total degree of the decryption circuit by more than a factor of 2 and thus the condition that the decryption protocol is admissible is maintained (and thus the scheme is bootstrappable. See the computation on p.18 [60]). Our approach for producing $\vec{s}$ is to securely generate a random number r_i in the range $[0, \Theta]$ for each s_i and setting $s_i = 1$ if $r_i \leq \theta$ and 0 otherwise.

Claim 24. *If each s_i is set to 1 with probability θ/Θ, then*

$$\Pr\left[\left|\sum_i s_i - \theta\right| > \theta/4\right] \leq 2^{-O(\lambda)}$$

Proof. Via the Chernoff bound. □

We also assume that the circular threshold security of the framework still holds with this modification. We believe that any natural proof in showing this modification still results in circular security would modify the original circular security argument for the base system. However, remember that the circular security of the original scheme is assumed, and therefore we cannot modify such a proof.

We set the parameter Θ to be a power of two to facilitate generating secret random elements smaller than Θ. In this case, generating secret random numbers smaller than Θ only requires concatenation of $\log \Theta$ secret random bits without any secret comparison. However in the general case, we would call the COMP protocol which is a relatively expensive operation. Therefore the algorithm to compute $\vec{s}$, presented in Figure 5.8, is as follows: For each s_i, the players produce shares of $\log \Theta$ random bits in step 1 (notice that the concatenation of these bits would be guaranteed to be in the interval $[0, \Theta]$). After local computation in step 2, the players securely compare the

result against θ to generate shares of s_i (i.e., s_i would be 1 with probability θ/Θ, and 0 otherwise). Also we set the last bit of the $\vec{s}$ to be 1. There reason is that it would simplify the next steps in the key generation protocol. Notice that setting the last bit to 1 would not change the security properties of the scheme. That is intuitively because there is polynomial chance that the last bit of the vector $\vec{s}$ was going to be set to 1, and also the adversary has polynomial chance of guessing it. Therefore any adversary that would break the modified scheme can be used to make an adversary that has also non-negligible chance in breaking the original scheme.

Protocol 3 . $[^\star\vec{s}], \theta' \leftarrow \mathrm{ComputeS}(\theta, \Theta)$

1: For $i = [\Theta]$ and $j = [\log \Theta]$, run $[^\star a_{i,j}] \leftarrow \text{RAN}_2()$
2: For $i = [\Theta]$, let $[^\star a_i]_B$ be $([^\star a_{i,\log \Theta}], \ldots, [^\star a_{i,1}])$
3: For $i = [\Theta]$, run Θ parallel executions of protocol $[^\star s_i] \leftarrow \text{COMP}([^\star a_i]_B, \theta)$
4: set s_Θ to 1
5: Locally compute $[^\star\theta'] \leftarrow \sum_i [^\star s_i]$
6: Run the protocol $\theta' \leftarrow \text{REVEAL}([^\star\theta'])$
7: Output $[^\star\vec{s}], \theta'$

Figure 5.8: A PROTOCOL TO SHARE $\vec{s}$

Complexity Analysis The algorithm produces $\Theta \log \Theta$ random bits in parallel, and performs another Θ comparisons in parallel (since a_i's length is at most $\log \Theta$ bits, therefore the produced value does not have to be compare against all ℓ bits, and hence the complexity is lower). Therefore it needs $2\Theta \log \Theta + \Theta(13 \log(\Theta) + 6\sqrt{\log(\Theta)})$ invocations of the multiplication protocol and 7 rounds of interaction.

Simulation. The simulator $S_{\mathrm{S}}(\theta, \Theta)$ for the protocol $\mathrm{ComputeS}$ works as follows:

1. Run the sub-simulator $S_{\text{RAN}_2}(0)$ for each call of RAN_2 protocol and obtain the adversary's share. Next, do the local computation in step 2 and obtain $[^\star\vec{a}]^I_B$.

2. For all $i \in [\Theta]$ run the sub-simulator $S_{\text{COMP}}([^\star a_i]^I_B, \theta, 0)$, and obtain the adversary's share $[^\star\vec{s'}]^I$.

3. Locally compute $[^\star\theta']^I = \sum_i [^\star s'_i]^I$.

4. Generate a random value θ'' with the same distribution as θ' in an honest execution.

5. Run $S_{\text{REVEAL}}([^\star\theta']^I, \theta'')$.

6. Output $[^\star\vec{s'}]^I$ and θ''.

Computing x_p and $\lfloor \frac{2^\kappa}{p} \rceil$

The secret key p for the "somewhat homomorphic encryption scheme" is an odd η-bit integer. To sample p, we notice that the bits p_0 and $p_{\eta-1}$ should be 1 whereas the rest of the bits $p_1, \ldots, p_{\eta-2}$ should be generated by having the players execute $\text{RAN}_2()$. Therefore, the secret key would be $2^{\eta-1} + \sum_{i=1}^{\eta-2} p_i 2^i + 1$ (which can be computed locally by players from shares of the bits). At the end, each player holds a share of p. The computation complexity involves $2(\eta-2)$ multiplication invocations and 2 rounds of interaction. The simulator for this subprotocol, named $S_{\text{MOD}}(p')$, is defined by calling $S_{\text{RAN}_2}(p'_i)$ for $i \in [\eta-1]$. The simulator outputs $[^\star p']^I$.

In [78], the authors present a method for two honest-but-curious parties to compute the average of their inputs. We extend their technique to allow multiple parties who hold shares of p to compute shares of $1/p$, and address the malicious model.We generalize [78]'s approach to calculate $\lfloor 2^\kappa/p \rceil$ by computing the first κ bits of $1/p$ and then rounding.

The approach works by calculating the first κ terms of the Taylor series of $1/p$. We can then bit-decompose the result and compute the desired result. Next, we determine the order of the field used for secret sharing and present the protocol and analysis.

Recall that p is subject to the constraint $2^{\eta-1} \leq p < 2^\eta$; set $\epsilon \in [0, 1/2]$ such that $p = 2^\eta(1-\epsilon)$. Thus:

$$p^{-1} = 2^{-\eta} \cdot \frac{1}{(1-\epsilon)} = 2^{-\eta} \sum_{i=0}^{\infty} \epsilon^i = 2^{-\eta} \left(\sum_{i=0}^{d} \epsilon^i \right) + 2^{-\eta} R_d$$

where $0 \leq R_d < 2^{-d}$. Multiplying both sides by $2^{\eta(d+1)}$ yields

$$2^{\eta(d+1)} p^{-1} = \left(\sum_{i=0}^{d} (2^\eta \epsilon)^i 2^{\eta(d-i)} \right) + \left(2^{\eta d} R_d \right) \tag{5.1}$$

Notice that $2^\eta\epsilon$ is an integer (since $p = 2^\eta(1-\epsilon)$ is an integer).

Let Z denote the first summand in 5.1 (i.e., $\sum_i (2^\eta\epsilon)^i 2^{\eta(d-i)}$). Having shares of p, players compute $2^\eta\epsilon$ collaboratively using the formula $2^\eta\epsilon = 2^\eta - p$. Holding shares of $2^\eta\epsilon$ and using the protocol power for exponentiation, the players can now compute shares of Z.

Because the exact integer value of Z is desirable, we need to choose the field Z_l used in the secret sharing scheme to be large enough to ensure correctness. In order to determine the bit-length of the field we first determine the maximum value $Z = \sum_{i=0}^{d} (2^\eta\epsilon)^i 2^\eta(d-i)$ can take. By 5.1 we have:

$$\sum_{i=0}^{d} (2^\eta\epsilon)^i 2^{\eta(d-i)} = 2^{\eta(d+1)} p^{-1} - 2^{\eta d} R_d$$

Our constraints ensure that $2^{\eta d} < 2^{\eta(d+1)} p^{-1} \leq 2^{\eta d+1}$ which immediately implies $Z \leq 2^{\eta d+1}$.

The constraints $0 \leq R_d < 2^{-d}$ imply that $\log(2^{\eta d} R_d) < \eta d - d$. But earlier we showed:

$$\eta d < \log(2^{\eta(d+1)} p^{-1}) \leq \eta d + 1$$

These two facts ensure that the error term $2^{\eta d} R_d$ will only change the least significant $\eta d - d$ bits in $2^{\eta(d+1)} p^{-1}$. The difference of the two bit lengths, $\eta d - (\eta d - d) = d$, is the number of bits that the error term does not change (assuming a carry will not happen). For our purposes, it suffices to compute the first κ bits of $1/p$ to yield $2^\kappa/p$. Therefore, we set $d = \kappa$.

Note that $[^\star Z]_{\kappa d \ldots \kappa d-\kappa}$ or $[^\star Z]_{\kappa d \ldots \kappa d-\kappa} + 1$ is the integer value of $\lfloor \frac{2^\kappa}{p} \rfloor$. However, we do not deal with rounding the result to the nearest integer. Instead, we can save the shares of the bit $Z_{\kappa d-\kappa-1}$ (which will determine if x_p should be added by 1 or not to have the desirable result) for later, and add it to dif in step 5 of the protocol in Figure 5.10. to make up the difference. This extra step would not affect the complexity of the protocol. In our protocol, we do not include this step for having simpler notation.

In the following protocol we formalize the above reasoning. We also set x_γ as the first γ bits of $[^\star Z]_{\eta\kappa \ldots \eta\kappa-\kappa}$. It is an output of the function. This value is needed later for generating the public key

$\langle x_0, \ldots, x_\tau \rangle$.

Protocol 4. $[^\star x_p] \leftarrow \text{ComputeXP}([^\star p], \kappa, \eta)$

1: Locally compute shares $[^\star e] \leftarrow [^\star 2^\eta] - [^\star p]$
2: Run the protocol $([^\star e], [^\star e^2], \ldots, [^\star e^\kappa]) \leftarrow \text{MULT}^*([^\star e], \kappa)$
3: Locally compute shares $[^\star Z] \leftarrow \sum_{i=0}^{\kappa} [^\star e_i] \cdot 2^{\eta(\kappa-i)}$
4: Run the protocol $[^\star Z]_B \leftarrow \text{BITS}([^\star Z])$
5: Locally compute $[^\star x_p] = [^\star Z]_{\eta\kappa..\eta\kappa-\kappa}$ by using $[^\star Z]_B$
6: Locally set $[^\star x_\gamma]_B = [^\star Z]_{\eta\kappa..\eta\kappa-\gamma}$
7: Output $[^\star x_p]$, $[^\star x_\gamma]_B$

Figure 5.9: A PROTOCOL TO SHARE x_p

Complexity Analysis In this protocol, the size of the field for secret sharing should be at least $2^{\eta\kappa+1}$. The above protocol invokes the MULT* subprotocol on line 2 and the BITS subprotocol on line 4 . Therefore these all multiplication invocation numbers adds up to: $31\ell \log \ell + 71\ell + 30\sqrt{\ell} + 6\kappa$. Round complexity analysis is as follows: line 2 takes 5 rounds and the line 4 takes 21 rounds (we can run the 2 preprocessing rounds in advance). The result is a total of $5 + 21 = 26$ rounds.

Simulation. The simulator for protocol 4, $S_{\text{XP}}([^\star p]^I, \kappa, \eta)$ is described as follows:

1. The first step of the protocol is a local computation.
2. For step 2, the simulator calls $S^*_{\text{MULT}}([^\star e]^I, \kappa, 0)$. At the end of this step, each player holds shares of 0 for each e_i and the simulator learns all the shares held by the adversary, $[^\star e_i]^I$.
3. For Step 3 is local computation. The simulator maintains knowledge of the shares of $[^\star Z]^I$.
4. For Step 4, the simulator calls $S_{\text{BITS}}([^\star Z]^I, \vec{0})$ and learns $[^\star Z]^I_B$.
5. Output $[^\star x_p]^I$ and $[^\star x_\gamma]^I_B$.

Producing $\vec{y}$ and $\vec{u}$

In [60], the vector $\vec{y}$ is generated as follows: 1) Sample integers $u_i \in Z \cap [0, 2^{\kappa+1})$, $i \in [\Theta]$ such that $\sum_i u_i \cdot s_i = x_p \mod 2^{\kappa+1}$, and 2) For $i \in [\Theta]$, set $y_i = u_i/2^\kappa$ (using κ bits of precision). Next,

we present a protocol $\mathrm{ComputeY}$ in Figure 5.10. for computing $\vec{y}$ and $\vec{u}$. This step requires a few relaxations to the FHE scheme to allow for more efficient implementation.

Since $\vec{u}$ reveals all information in $\vec{y}$ and nothing more, it suffices to compute and reveal $\vec{u}$. To compute $\vec{u}$, we generate random numbers in the interval $[0, 2^{\kappa+1})$ for each u_i in lines $1-3$ (no information is revealed since the generated values are random integers). Next, we need to modify the vector $\vec{u}$ to satisfy the constraint:

$$\sum_i u_i \cdot s_i = x_p \bmod 2^{\kappa+1}$$

Remember that we set the vector $\vec{s}$ so that s_Θ would be one. Therefore, we can produce all elements in $\vec{u}$ randomly, secretly multiply them by $\vec{s}$, and reveal the result. Based on this result, we can modify the value of u_Θ to satisfy the constraint. In more details, we replace the u_Θ with $2^{\kappa+1}-1$ (to make sure the result of subtraction in step 5 would not be negative), compute $dif = x_p - \sum_i u_i \cdot s_i \mod 2^{\kappa+1}$, and reveal the result. The users, then, can locally replace u_Θ with $u_k - dif \mod 2^{\kappa+1}$ in step 7 which guarantees that $\sum_i u_i \cdot s_i = x_p \mod 2^{\kappa+1}$.

Protocol 5. $\vec{y} \leftarrow \mathrm{ComputeY}([^\star x_p], [^\star \vec{s}])$

1: For $i \in [\Theta]$, run protocol $[^\star u_i] \leftarrow \textsc{ran}_p()$ in parallel
2: For $i \in [\Theta]$, run protocol $u_i \leftarrow \textsc{reveal}([^\star u_i])$
3: For $i \in [\Theta]$, locally set $u_i \leftarrow u_i \mod 2^{\kappa+1}$
4: Locally set $u_\Theta \leftarrow 2^{\kappa+1}$
5: $[^\star dif] \leftarrow (\sum_i u_i \cdot [^\star s_i] - [^\star x_p])$
6: Run protocol $dif \leftarrow \textsc{reveal}([^\star dif])$
7: Locally compute $u_\Theta \leftarrow u_\Theta - (dif \mod 2^{\kappa+1})$
8: Output $\vec{u}/2^\kappa$.

Figure 5.10: A Protocol to Share $\vec{y}$

Complexity Analysis. $\mathrm{ComputeY}$ only calls $\textsc{ran}_p$ Θ times in line 1. Hence the protocol needs 1 round of multiplication interactions, and Θ number of multiplication invocation. The largest value computed is upper bounded by $\theta \cdot (2^{\kappa+1})$ in line 5, therefore ℓ needs to have a bit-length of at least $\log \theta \cdot (\kappa + 1)$.

Simulation. The simulator $S_{\mathrm{Y}}([^{\star}x_p]^I, [^{\star}\vec{s}]^I, \vec{y'})$ acts as follows:

1. Let $\vec{u'} = \vec{y'} \cdot 2^{\kappa}$,
2. For the first step, the simulator calls the simulator $S_{\mathrm{RAN}_p}(0)$ for $i \in [\Theta]$ and learns $[^{\star}u_i]^I$,
3. The simulator calls $S_{\mathrm{REVEAL}}([^{\star}u_i]^I, u'_i)$ for $i \in [\Theta - 1]$. Generate a random $u''_{\Theta} \in [p]$ and call $S_{\mathrm{REVEAL}}([^{\star}u_{\Theta}]^I, u''_{\Theta})$.
4. The simulator follows steps 3 to 5,
5. Set dif' to $(2^{\kappa+1} - u'_{\Theta})$. The simulator calls $S_{\mathrm{REVEAL}}([^{\star}dif]^I, dif')$,
6. The simulator follows steps 7 to 8.

Computing $\langle x_0, \ldots, x_{\tau} \rangle$

Recall from the original public-key generation algorithm that we need to sample $x_i \leftarrow D_{\gamma,\rho}(p)$ for $i = 0, \ldots, \tau$. Intuitively, these x_i represent random encryptions of 0 that get added to our base encryption in the homomorphic scheme. Further, recall that

$$D_{\gamma,\rho}(p) = \{\text{choose } q \leftarrow Z \cap [0, 2^{\gamma}/p), r \leftarrow Z \cap (-2^{\rho}, 2^{\rho}) : \text{ output } x \leftarrow pq + r\}.$$

After sampling, the list should be relabeled so that x_0 is the largest. The key-generation process requires that the process is restarted if either x_0 is even or $x_0 - \lfloor x_0/p \rceil \cdot p$ is odd. Since $x_0 = pq + r$ is generated as directed for some random q and r and since p is an odd number, the requirement that x_0 is odd can be checked by inspecting the least significant bits of the q and r: If $q_0 + r_0 = 1$, then x_0 satisfies the first condition.

To check the second condition, that $x_0 - \lfloor x_0/p \rceil \cdot p$ is an odd number, we observe that because of the constraints $-2^{\rho} < r < 2^{\rho}$ and $2^{\eta-1} \leq p < 2^{\eta}$, it follows that

$$-2^{\rho-\eta+1} < r/p < 2^{\rho-\eta+1}$$

Since $\rho = \lambda$ and $\eta = \tilde{O}(\lambda^2)$, therefore for all sufficiently large λ (if $\eta = \lambda^2$, then for $\lambda > 2$), $\lfloor r/p \rceil = 0$ and as a result r can be ignored. That is $\lfloor x_0/q \rceil = \lfloor pq + r/q \rceil = q + \lfloor r/q \rceil = q$. So $x_0 - \lfloor x_0/p \rceil \cdot p = x_0 - q \cdot p$. Because x_0 and p are both odd, q must be odd to make the term $x_0 - \lfloor x_0/p \rceil \cdot p$ even. These constraints imply that for x_0 to be odd and $x_0 - \lfloor x_0/p \rceil \cdot p$ to be even, then q must be even and r must be odd.

To sample $q \in [0, 2^\gamma/p)$, we first compute $\lfloor 2^\gamma/p \rfloor$. The bit decomposition of $\lfloor 2^\kappa/p \rfloor$ (or potentially the bit decomposition of $\lfloor 2^\kappa/p \rfloor - 1$, but it does not matter since it makes negligible difference) from the protocol ComputeXP as $[x_\gamma]_B$ can be used to compute $\lfloor 2^\gamma/p \rfloor$. We modify the SOLVED-BITS algorithm to return the least significant bit of the the value at no extra cost. Also since the least significant bits for both q and r associated with x_0 are random values, the chance of the algorithm not aborting is $1/4$. Therefore, if we need a constant round algorithm we need to run the algorithm λ times in parallel to ensure that the chance of aborting in all runs negligible.

The algorithm for producing PK is presented in Figure 5.11 (within, we refer to $\lfloor 2^\gamma/p \rfloor$ as $[x_\gamma]_B$).

Protocol 6 . $\langle x_0, \ldots, x_\tau \rangle \leftarrow \text{ComputeX}([^\star x_\gamma]_B, [^\star p], \tau, \rho)$

1: For $i \in [\tau]$ run protocol $[^\star q_i], [^\star q_{i,0}] \leftarrow$ SOLVED-BITS$([^\star x_\gamma]_B)$ in parallel
2: For $i \in [\tau]$ and for $j = 0..\rho$, run protocol $[^\star r_{i,j}] \leftarrow$ RAN$_2()$ in parallel
3: For $i \in [\tau]$, locally compute $[^\star r_i] \leftarrow (\sum_{j=0}^{\rho-1} [^\star r_{i,j}] 2^j) \cdot (2 \cdot [^\star r_{i,\rho}] - [^\star 1])$
4: For $i \in [\tau]$, run $[^\star x_i] \leftarrow [^\star p] \cdot [^\star q_i] + [^\star r_i]$ in parallel
5: For $i \in [\tau]$, run protocol $x_i \leftarrow$ REVEAL$([^\star x_i])$ in parallel
6: $x_0 \leftarrow$ biggest of the revealed x_i's
7: Reveal the least significant bits from q and r in computing x_0. If either the former is not even, or the latter is not odd, abort
8: Output $\langle x_0, \ldots, x_\tau \rangle$

Figure 5.11: A PROTOCOL TO SHARE $\vec{x}$

Complexity Analysis Lines 1 and 2 can be run in parallel, and require 8 rounds and $\tau(52\gamma + 24\sqrt{\gamma} + 2\rho)$ multiplications (since x_γ is a shared value, the complexity is slightly higher. Also note that $[x_\gamma]_B$'s length is γ bits, therefore the produced value does not have to be compare against all ℓ bits, and hence the complexity is lower). We need 2 multiplications in lines 3 and 4, but these can be done in parallel. Hence, the total round complexity would be 9, and the total number of

multiplication invocations would be $\tau(52\gamma + 24\sqrt{\gamma} + 2\rho + 2)$. The largest number used in this protocol has at most γ bits, so the field order needs at least γ bits.

Simulation. For simplicity, we only show one execution of the simulator for the ComputeX protocol. Recall that if the least significant bits of q and r are not respectively even and odd, the algorithm should abort. Obviously these values for each run are public, and hence can be given as input to the simulator for as many times as it takes to get the final $\vec{x}$. For simplicity, we omit these values from the input of the simulator. The simulator $S_{\mathrm{X}}([^\star e]^I, [^\star p]^I, \vec{x'}, \tau, \rho)$acts as follows:

1. The simulator calls $S_{\mathrm{SOLVEDBITS}}([^\star e]^I, 0)$ and learns $[^\star q_i]^I$,
2. The simulator calls $S_{\mathrm{RAN}_2}(0)$ for step 2 and learns $[^\star r_i]^I$,
3. The simulator calls $S_{\mathrm{MULT}}([^\star p]^I, [^\star q]^I, 0)$. Knowing $[^\star r_i]^I$, the simulator learns $[^\star x_i]^I$,
4. The simulator calls $S_{\mathrm{REVEAL}}([^\star x_i]^I, x'_i)$,
5. The simulator follows the step 6,
6. The simulator runs $S_{\mathrm{REVEAL}}([^\star q_0]^I, q_0)$ and $S_{\mathrm{REVEAL}}([^\star r_0]^I, r_0)$,
7. Output $\vec{x'}$.

Computing encryptions of $\vec{s}$

One step in Gentry's paradigm for FHE construction requires the public key to contain an encryption of the secret key. We assume circular security of the underlying encryption scheme, as do van Dijk et al. [60] and Gentry [79]. Towards this goal, we design a protocol that enables players who hold private shares of the secret key (as well as the entire public key) to compute an encryption of the secret key under the public key. Note this *cannot* be done trivially with homomorphic evaluation because the encrypted secret-key is in fact necessary to homomorphically evaluate circuits of an arbitrary depth, resulting in a circular requirement. Similar issues arise when trying to produce a public-key for a leveled fully homomorphic encryption scheme.

Recall that in Dijk et al. [60], the encryption of m under the public key $\langle x_0, \ldots, x_\tau \rangle$ computes as $[m + 2r + 2\sum_{i \in S} x_i]_{x_0}$, where $r \in (-2^{\rho'}, 2^{\rho'})$ and $S \subseteq \{1, \ldots, \tau\}$ is a random subset. Since both the x_i's and r can take negative values (as integers) whereas the computation is in a finite field, we need to somehow make sure the computation in the finite field result in the same integer value of the encryption of m. To resolve this issue, we compute the value min which is a unique value that satisfies the following two properties: 1) $min = 0 \mod x_0$, and 2) for an arbitrary S and for our set of x_i's and any value of r, it would make the summation $m + 2r + 2\sum_{i \in S} x_i$ positive. Because the range of values that r can take is public, all users can compute min locally and agree on respective shares. Next, to encrypt the secret key, all users generate shares for a set S and the shares for a value r. All users then add their shares of r, use shares in S to add in appropriate x_i's, and add min.

The players run the protocol presented in Figure 5.12 for each of the s_i's to obtain its encryption.

Protocol 7 . $(c_k) \leftarrow \text{EncryptS}([^\star s_k], \langle x_0, \ldots, x_\tau \rangle)$

1: Locally compute min as directed
2: For $i = 0..\rho'$ run the protocol $[^\star r_i] \leftarrow \text{RAN}_2()$ in parallel
3: For $i \in [\tau]$ run the protocol $[^\star S_i] \leftarrow \text{RAN}_2()$ in parallel
4: Locally compute $[^\star r] \leftarrow \sum_{i=0}^{\rho'-1} [^\star r_i] 2^i$
5: Run the protocol $[^\star r] \leftarrow [^\star r] \cdot (2 \cdot [^\star r_{\rho'}] - [^\star 1])$
6: Locally compute $[^\star c'_k] \leftarrow [^\star s_k] + [^\star r] + 2\sum_i [^\star S_i] \cdot x_i + [^\star min]$
7: Run the protocol $c'_k \leftarrow \text{REVEAL}([^\star c'_k])$
8: Locally compute $c_k \leftarrow c'_k \mod x_0$
9: Output c_k

Figure 5.12: A PROTOCOL TO COMPUTE THE ENCRYPTION OF $\vec{s}$

Complexity Analysis The protocol produces bits in lines 2 and 3 which take 2 rounds of interaction (they can be run in parallel) and a total of $2(\rho' + \tau + 1)$ multiplication invocations. We perform another multiplication in line 5. Therefore, the protocol needs 3 rounds of interaction and invokes the multiplication protocol $(2(\rho' + \tau + 1) + 1)\Theta$ times (the whole term gets multiplied by Θ because we need to encrypt all s_i's).

Simulation. For $i \in [\Theta]$, the simulator $S_{\text{ENCS}}([^\star s_i]^I, \vec{x}, c'_i)$ acts as follows:

1. For steps 2 and 3, the simulator calls $S_{\text{RAN}_2}(0)$, and learns $[^\star r_i]^I$ and $[^\star S_i]^I$,
2. For step 5, the simulator calls $S_{\text{MULT}}([^\star r]^I, (2 \cdot [^\star r_{\rho'}] - [^\star 1])^I, 0)$, and learns $[^\star r]^I$,
3. Having shares of $[^\star s_k]^I$, and $[^\star r]^I$, the simulator locally computes $[^\star c'_k]^I$,
4. For step 7, the simulator calls $S_{\text{REVEAL}}([^\star c'_k]^I, c'_i)$,
5. For step 8, locally compute $c_i = c'_i \mod x_0$,
6. Output c_i.

Computing encryptions of 0 and 1 for PK

The same techniques from the previous step can be used to produce encryptions of random bits. These encryptions can then be collaboratively decrypted until both an encryption of 0 and an encryption of 1 are identified. These two ciphertexts can then be adjoined to the public key—they are guaranteed to be well-formed and thus have the right amount of noise.

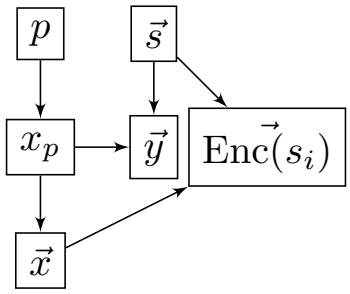

Figure 5.13: THE COMPLETE KEY GENERATION PROTOCOL HIERARCHY

Complete key generation protocol

We now put all of the pieces together and describe the entire key generation protocol. As we mentioned earlier, the public key consists of $\langle x_0, \ldots, x_\tau \rangle$, $\vec{y}$, and $\forall_i \vec{\textbf{Enc}}(s_i)$ which can be instantiated by calling the protocols ComputeX, ComputeY, and EncryptS. The secret key is the vector $\vec{s}$ which can be instantiated by calling the protocol ComputeS. To compute these values, we introduced two other helper protocols to generate the values p and x_p. For these two, we need to call protocols

	rounds	mult	field
p	2	$O(\eta)$	η
$\vec{s}$	7	$O(\Theta \log \Theta)$	$\log \Theta + 1$
x_p	26	$O(\ell \log \ell)$	$\eta\kappa + 2$
$\vec{x}$	9	$O(\tau\gamma)$	γ
$\vec{y}$	1	$O(\Theta)$	$\lceil \log \theta(\kappa+1) \rceil + 1$
$\mathbf{Enc}(s_i)$	3	$O(\Theta\tau)$	$2(\log \tau)\gamma$

Table 5.1: COMPLEXITY ANALYSIS OF KEY GENERATION SUB-PROTOCOLS

ComputeP and ComputeXP. Figure 5.13 shows the calling sequence for these protocols. The resulting protocol to generate $(\text{PK}, \text{SK}_1, \ldots, \text{SK}_n)$ (where n is the number of players) is presented in Figure 5.14.

Protocol 8 . $(\text{PK}, \text{SK}_1, \ldots, \text{SK}_n) \leftarrow \tilde{\text{G}}(\eta, \tau, \rho, \theta, \Theta, \kappa)$

1: Call protocol $[^\star p] \leftarrow \text{ComputeP}(\eta)$.
2: Call protocol $[^\star \vec{s}], \theta' \leftarrow \text{ComputeS}(\theta, \Theta)$
3: Call protocol $[^\star x_p], [^\star x_\gamma]_B \leftarrow \text{ComputeXP}([^\star p], \kappa, \eta)$
4: Call protocol $\vec{x} \leftarrow \text{ComputeX}([^\star x_\gamma]_B, [^\star p], \tau, \rho)$
5: Call protocol $\vec{y} \leftarrow \text{ComputeY}([^\star x_p], [^\star \vec{s}])$
6: For $i \in [\Theta]$, call protocol $c_i \leftarrow \text{EncryptS}([^\star s_i], \vec{x})$
7: Generate public ciphertexts c_0, c_1 corresponding to 0 and 1
8: Output $\text{PK} = (\langle x_0, \ldots, x_\tau \rangle, \vec{y}, \vec{c}, c_0, c_1)$, and $\text{SK}_1, \ldots, \text{SK}_n$ where SK_i is player i's share of $\vec{s}$

Figure 5.14: THE COMPLETE KEY GENERATION PROTOCOL

Complexity Analysis Table 5.1 gives a summary of the round and multiplication complexity of each of the subprotocols. *If we run non-sequential protocols in parallel, the total number of rounds $\tilde{G}$ needs is 40.*

Also the last column in Table 5.1 represents the minimum bit-length of ℓ (the field order) for each of the sub-protocols. The maximum of all these values is $\eta\kappa + 2$, which we can substitute as ℓ in all equations. Knowing ℓ, the multiplication invocation number would be the summation of the second column.

Simulation The simulator $S_{\tilde{G}}(\text{PK} = (\langle \vec{x'} \rangle, \vec{y'}, \vec{c'}), \eta, \tau, \rho, \theta, \Theta, \kappa)$:

1. The simulator chooses a random p' which is an odd η-bit integer, and runs $S_{\mathrm{P}}(p', \eta)$ to learn $[^\star p']^I$,
2. The simulator runs $S_{\mathrm{S}}(\theta, \Theta)$ and learns $[^\star \vec{s'}]^I$ and θ',
3. The simulator runs $S_{\mathrm{XP}}([^\star p']^I, \kappa, \eta)$ and learns $[^\star x_p]^I$ and $[^\star x_\gamma]^I$,
4. The simulator runs $S_{\mathrm{X}}([^\star x_\gamma]^I, [^\star p']^I, \vec{x'}, \tau, \rho)$,
5. The simulator runs $S_{\mathrm{Y}}([^\star x_p]^I, [^\star \vec{s'}]^I, \vec{y'})$,
6. For each $i \in [\Theta]$, the simulator calls $S_{\mathrm{ENCS}}([^\star \vec{s'}]^I, \vec{x'}, c'_i)$,
7. Output $(\mathrm{PK}, [^\star \vec{s'}]^I)$.

5.4.4 Constant Round Decryption

In this section, the notation $[a]_2$ means $(a \mod 2)$ and $\overline{a}$ means $(1 - a)$. We assume the field order in which secrets are shared is an integer p. Now recall the encryption algorithm.

Enc$(\mathbf{PK}, m)$ Generate a ciphertext c^* as before in **Enc**$'$ (see below). Then for $i \in 1, \ldots, \Theta$, set $z_i \leftarrow [c^* \cdot y_i]_2$, keeping only $\lceil \log \theta \rceil + 3$ bits of precision after the binary point for each z_i. Output both c^* and $\vec{z} = \langle z_1, \ldots, z_\Theta \rangle$.

Enc$'(\mathbf{PK}, m \in \{0, 1\})$ Choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$ and a random integer r in $(-2^{\rho'}, 2^{\rho'})$. Output $c^* \leftarrow [m + 2r + 2\sum_{i \in S} x_i]_{x_0}$.

We observe that for an arbitrary $\zeta = 2^{-z}$ for some integer z, there is a specific set of the FHE scheme's parameters that make $\sum_i s_i z_i$ within $\zeta < 1$ of an integer (in our setting, ζ is $1/4$). Therefore, by basic properties of addition mod 2, we have:

$$\left[c^* - \left\lfloor \sum_i s_i z_i \right\rfloor\right]_2 = \left[[c^*]_2 - \left[\left\lfloor \sum_i s_i z_i \right\rfloor\right]_2\right]_2 = \left[[c^*]_2 - \left[\left\lfloor \sum_i s_i z_i + \zeta \right\rfloor\right]_2\right]_2$$

Since c^* is public, the decryption can be determined by revealing the first bit after binary point (i.e., the decimal point in a binary number) in $\sum_i s_i z_i + \zeta$. Assuming each party P_j has shares of

$[^\star s_i]$, the value $\sum_i s_i z_i + \zeta$ can be computed locally and revealed. But revealing bits other than the first bit after binary point in $\sum_i s_i z_i + \zeta$ might leak information about the secret key. Hence we need to distort all bits that are not relevant to the final answer.

In order to re-randomize all of the other bits that happen to be revealed, we add a random value r to $x = (\sum_i s_i z_i + \zeta)2^{k-1}$, where $k = \log\theta + 3$ (the multiplication by 2^{k-1} removes the binary point). We choose the field order p that we secret share in such that $0 \leq x_{max} < p - 2^{|p|-1}$ (x_{max} is the maximum value x can take and is equal to $2 \cdot \theta \cdot 2^{\lceil \log\theta + 3 \rceil}$). The reason for choosing p in this way is to determine if a wrap-around happens in $x + r$ for some random number r based on the most significant bit of r and the result. More precisely, if the result of summation's most significant bit is 0 and if the most significant bit of r is one, then a modular reduction (i.e., wrap-around) occurred, and the result should be added to p. In step 7 we decide if such event occurred or not and we determine the integer value of the summation of $x + r$ in step 8.

Given the integer value of $x + r$ and knowing that the $(k-1)^{\text{th}}$bit of the result is 0, the k^{th}bit can be calculated as $\left[R''_k - r_k - r_{k-1}\overline{R''_{k-1}}\right]_2$. Steps 9 to 12 calculate this value securely.

Let $|z_i| = \log\theta + 4 = k$. The protocol for multiparty decryption of a ciphertext is presented in Figure 5.15.

Protocol 9 . $m' \leftarrow \tilde{D}(c^*, [^\star s_1], \ldots, [^\star s_n])$

1: Players locally compute $\vec{z}$ as directed. Let $\vec{z'} = 2^{k-1} \cdot \vec{z}$
2: Run protocol$([^\star r]_B, [^\star r]) \leftarrow$SOLVED-BITS$()$
3: Locally compute $[^\star x] \leftarrow \sum_i [^\star s_i] \cdot z'_i$
4: Locally compute $[^\star R] \leftarrow [^\star x] + [^\star r]$
5: Run protocol $R \leftarrow$ REVEAL$([^\star R])$
6: Locally compute $R' \leftarrow R + p$
7: Locally compute $[^\star c] \leftarrow [^\star r_{\ell-1}] \cdot \overline{R_{\ell-1}}$
8: For $i = 0, ..., k$ locally compute $[^\star R''_i] \leftarrow [^\star c] \cdot R'_i + ([^\star 1] - [^\star c]) \cdot R_i$ in parallel
9: Locally compute $[^\star a] \leftarrow [^\star R''_k] - [^\star r_k]$
10: Run protocol $[^\star a'] \leftarrow [^\star a^2]$
11: Locally compute $[^\star a''] \leftarrow [^\star a'] - [^\star r_{k-1}] \cdot \overline{R''_{k-1}}$
12: Run protocol $[^\star a'''] \leftarrow [^\star a''^2]$
13: Run protocol $a''' \leftarrow$ REVEAL$[^\star a''']$
14: Output $m' \leftarrow [c^* - a'''_2]$

Figure 5.15: THE DECRYPTION PROTOCOL

Notice that setting p as a Mersenne Prime decreases the round complexity. This is because the binary digits of such a prime are all 1. In step 2, the players collaborate on producing a random number in field p. The most expensive part of this step is to check if $r < p$ which takes 7 rounds. If p is Mersenne, we do not need such a check because the only case the produced r is not less than p is when r is equal to p.

Claim 25. ***Dec***$(sk, c^*) = [c^* - a''']_2$.

Proof. By the assumed setup, each player p_i holds the shares of $[^\star s_i]$'s for all $0 \leq i \leq \Theta$. Recall that

$$\textbf{Dec}(c^*, z) = \left[c^* - \left\lfloor \sum_i s_i z_i \right\rfloor \right]_2$$

and that $k = \log \theta + 3$ and $x = \lfloor \sum_i s_i z_i \rceil \, 2^k$. All we need to prove is that $x_{k+1} = a'''$. Since c^* is public, revealing either x_{k+1} or m' would result in determining the other one.

By definition we have:

$$x \leq \theta \cdot 2^{k+2},$$

and we have

$$0 \leq x < p - 2^{|p|-1}$$

by the selection of parameter p. Instruction 2 defines a value $r \in [0, p]$ that is shared among the n players and instruction 4 defines $R \leftarrow x + r \bmod p$. Therefore, we have:

$$R'' = (x + r) = \begin{cases} R' = R + p & \textit{if } r > 2^{|p|-1} \text{ and } R < 2^{|p|-1} \\ R & \textit{o.w} \end{cases}$$

When $r > 2^{|p|-1}$ and $R < 2^{|p|-1}$, it follows that $r_{|p|} = 1$ and $R_{|p|} = 0$ (i.e. the high-order bits of the values of r and R are 1 and 0 respectively). We conclude:

$$[^\star R'']_B = [^\star r_{|p|}] \cdot \overline{R_{|p|}} \cdot R' + \overline{[^\star r_{|p|}]} \cdot R_{|p|} \cdot R$$

We have shown that R'' is the integer value of $x + r$. It is left to determine if a carry occurs between the bits k and $k + 1$ in the addition $x + r$. If no carry happens, a''' is $\left[{}^{\star}R''_{k+1} - r_{k+1}\right]_2$. If a carry between the mentioned bits happens, a''' is $\left[{}^{\star}R''_{k+1} - r_{k+1} + 1\right]_2$. Let a_{b-0} denote the substring $a_b...a_0$. To determine if a carry happens between the bits k and $k + 1$, we consider the following cases:

1. Case 1: $[{}^{\star}r]_{k,0} < 2^k$. As we mentioned earlier, $0 \leq [{}^{\star}x]_{k,0} < 2^k$. Therefore if $x + r < 2^{k+1}$, then no carry occurs.

2. Case 2: $2^k \leq r_{k-0} < 2^{k+1}$. So:

$$2^k < x_{k-0} + r_{k-0} < 2^{k+1} + 2^k$$

   The upper limit in the above equation means that both $k+1$th and kth bits of integer summation $x_{k-0} + r_{k-0}$ cannot be 1 at the same time. On the other hand, the lower limit guarantees that at least one of these bits is 1. Therefore, in this case a carry happens if and only if $R''_k = 0$.

Combining these results we conclude that $r_k\overline{R''_k}$ is 1 if a carry happens, and is 0 otherwise. Therefore:

$$a'''_0 = \left[\left[{}^{\star}R''_{k+1}\right] - \left[{}^{\star}r_{k+1}\right] - \left[{}^{\star}r_k\right] \cdot \left[{}^{\star}\overline{R''_k}\right]\right]_2$$

Instructions 9 to 12 computes the following value:

$$a''' = \left(\left(R''_{k+1} - r_{k+1}\right)^2 - r_k\overline{R''_k}\right)^2$$

Finally, we need to prove $a'''_0 = a'''$. A truth table verifies the equality. Also using truth tables it is easy to see that a''' reveals either 0 or 1 and no other value, (otherwise it might leak information regarding the variables r_{k+1}, $\overline{R''_k}$, R''_{k+1}, or r_k). □

Complexity Analysis The protocol calls the SOLVED-BITS subprotocol in line 2 and calls multiplication in lines 10 and 12. Therefore, the protocol takes 9 rounds and requires $(52|p| + 24\sqrt{p} + 2)$

multiplications. From the key generation section above, recall that the field order needs to be $\eta\kappa + 2$. This constraint is still sufficient, as the only constraint needed here is that $0 \leq x_{max} < p - 2^{|p|-1}$.

Simulation To prove security, we describe the required simulator S (ideal model adversary) that generates the view of real-life adversary. As usual, the simulator works by running an internal copy of the real-life adversary and an internal copy of the honest players. At a high level, the simulator extracts the shares of the real-life adversary, feeds this share to an ideal functionality which implements the k-th bit extractor function, and using the answer received, produces a view for the adversary which is consistent with the answer.

For adversary A, the simulator $S_{\tilde{D}}(c^*, [^\star s_1]^I, \ldots, [^\star s_\Theta]^I, b^*)$ on input the security parameter, the ciphertext c^*, the shares of secret s known by adversary, and $b^* = \textbf{Dec}(c^*, z)$ does the following:

1. For Step 2 of the protocol, the simulator $S_{\tilde{D}}$ runs the sub-simulator $S_{\text{SOLVEDBITS}}()$ to produce the output r and its resolved bits. The simulator also learns $[^\star r_i]^I$ and $[^\star r]^I$.

2. For steps 3- 4 of the protocol, the simulator follows the protocol on behalf of the honest players that it simulates.

3. The simulator picks a random $\hat{x}$ which satisfies the following constraints: $\hat{x}$ should be a random number in the range of 0 and M, $\hat{x}/2^{k-1} - 1/4$ should be within $1/4^{\text{th}}$of an integer, and $\hat{x}_k = b^*$ (i.e., the k^{th}bit of $\hat{x}$ should be b^*).

4. Using $\vec{z}$, $[^\star s_i]^I$, and $[^\star r_i]^I$, simulator computes the shares that the adversary would hold as the shares of R, $[^\star \vec{R}_i]^I$. The simulator then run the $S_{\text{REVEAL}}([^\star \vec{R}_i]^I, (\hat{x} + r))$ sub-simulator to reveal $\hat{x} + r$ to the adversary.

5. The simulator follows the steps 6- 14 of the protocol.

Correctness of Simulation The simulator and the real execution of the protocol only differ in steps 3 and 5 of the $\tilde{D}$ protocol. Since SOLVED-BITS and REVEAL protocols are secure protocols

their simulations are indistinguishable from the real world executions. In step four, instead of outputting the real $x + r$, we reveal $\hat{x} + r$. Since r is a totally random number, therefore the output will be a random number as well and indistinguishable. Since $\hat{x}$ is taken from the same domain as x would be taken, therefore the result has the same correctness properties as $x + r$. The rest of the protocol is identical in both scenarios. As a result, the view of the adversary in a real execution is identical to view of the adversary in simulated execution.

Outputs of All Players. It is left to show that the output computed by all players in a real execution and in an ideal execution with an ideal adversary are identically distributed. The output of the simulation is identical to the output of the real execution, since the $\tilde{D}$ protocol reveals the $\log +4^{\text{th}}$bit of x. As a result if this bit being set to be the same as the expected decrypted value, the output would be identical. Since the simulator has full control over setting the value of x the output will be identical to that of the real execution.

Theorem 5.4.2. *(Informal) The threshold scheme* $\Pi = (\tilde{G}, \tilde{E}, \tilde{D}, \tilde{\textbf{\textit{Eval}}})$ *described above satisfies the Threshold Indistinguishability Security notion as per Definition 5.2.2 and the Circular Threshold Security notion as per Definition 5.2.3.*

Proof. (Proof Sketch) The indistinguishability and circular security for our scheme Π follows from the simulatability of the key generation procedure. We show how to transform an adversary A that has advantage ϵ in the Threshold indistinguishability game into an adversary A' for the FHE semantic security that also has advantage ϵ. The reduction is straightforward. Adversary $A'(1^k, pk)$, upon receiving a public key, runs the simulator for the keygen protocol with adversary A to begin an internal execution of the threshold security game. Adversary A eventually receives a set of C shares of a secret key (that are statistically independent of sk), and then produces a pair of messages. A' forwards these messages and then forwards the challenge ciphertext c^* to A, and finally echoes A's response as output. Notice that A' produces a statistical simulation of the threshold semantic security game owing to statistical security of the simulator for the threshold key generation procedure. Thus, the advantage of A' is also ϵ, and the security of the FHE scheme implies that ϵ must therefore

be negligible in the security parameter k. Next, observe that the circular threshold security of the scheme can be argued in the same fashion. □

5.5 Secure Multiparty Computation

We follow the Cramer et al. [68] approach for constructing a multi-party computation protocol based on threshold cryptography. Our biggest changes are that we do not need a protocol for multiplication, we use a different approach for proving knowledge of encryption, and we explicitly describe a key generation phase whereas it is assumed as an external setup in [68]. Since our solution requires less interaction among the parties, our simulation argument is simpler than the argument from [68].

We use the standard simulation-based definition of stand-alone secure multi-party computation. We assume the existence of a standard n-party CoinFlipping protocol which guarantees soundness in the presence of $< n/2$ adversaries: namely, for any minority set of adversaries, the protocol guarantees that the distribution is still statistically close to uniform. Such a protocol can be easily constructed based on the existence of hiding commitments. (Unlike CDN, we do not need this coin flipping protocol to be simulatable.). See §5.5.2 for a definition of the real/ideal paradigm for secure multi-party computation from [68] and [65]. In this section the TFHE scheme used is denoted $\tilde{\Pi} = (\tilde{G}, \tilde{E}, \tilde{D}, \textbf{Eval})$.

5.5.1 MPC Using Fully Homomorphic Encryption Scheme

In this section we present our protocol for evaluating an arbitrary circuit f in the presence of a minority of malicious adversaries. We assume that the players can communicate via an authenticated broadcast channel and via point-to-point private and authenticated channels (which may in turn be implemented using signatures, public key encryption, etc.)

5.5.2 Security

In this section we adopt a standard security definition for secure multi-party computation proposed by Canetti [80] and as used in [68] and [65]. This definitional approach compares the real-world execution of a protocol for computing a function with an ideal-world evaluation of the function by a trusted party. Security is then defined by requiring that for every adversary A attacking the

Protocol 10 . Each party holds private input x_i; the parties jointly compute $f(x_1, \ldots, x_n)$

1: Party P_i receives as input $(1^k, n, x_i)$. (We assume the adversary receives as input $1^k, n$, a set of corrupted parties C and the inputs $\{x_c\}_{c \in X}$ for the corrupted parties, and auxiliary information.)

2: Players run the key generation subprotocol $\tilde{G}(\eta, \tau, \rho, \theta, \Theta, \kappa)$ to generate a public key $\tilde{\text{PK}}$ and shares of the secret for the threshold scheme $\tilde{\Pi}$. At the end of this step, player p_i holds share $[s_i]$. If the sub-protocol halts prematurely, then players halt and output $\perp$.

3: The players take sequential turns sharing their input using the encryption scheme $\hat{\Pi}$ that is constructed from Π (see §5.3.4). More specifically, for $i \in [n]$, player P_i broadcasts $c_{i,j} \leftarrow \hat{E}(\tilde{\text{PK}}, x_{i,j})$. Then all of the players run a standard CoinFlipping protocol to generate a random string r_i. Player P_i now interprets r_i as n strings $r_{i,1}, \ldots, r_{i,n}$ and uses coins $r_{i,j}$ as the random coins to run Verifier(PK,$c_{i,j}$) (see §5.3.4) of the Hidden Bit POK protocol on input $c_{i,j}$ for each bit $j \in [n]$ of input x_i. Player P_i runs the corresponding Prover algorithm on $c_{i,j}$ using the random coins used to generate $c_{i,j}$ as the witness, and broadcasts the Prover message. The remaining players also execute the Verifier algorithm using the same random coins and verify that the first message is consistent and the second message is accepted. If player P_i fails the POK protocol, then P_i is excluded from the rest of the protocol, and the remaining players that have not been excluded use a canonical encryption of 0 as the input for P_i (e.g., they use $\tilde{E}(\tilde{\text{PK}}, 0; 0)$ as each input bit).

4: The players that have not been excluded locally run **Eval**$(\tilde{\text{PK}}, c_{1,1}, \ldots, c_{n,n}, \tilde{f})$ where the function $\tilde{f}$ first transforms the input ciphertexts encrypted under $\hat{\Pi}$ into ones for scheme $\tilde{\Pi}$. This is done by homomorphically evaluating the decryption procedure described in §5.3.3 (i.e. bootstrapping, see Definition 5.2.5).(Note: All of the ciphertexts in $c_{i,j}$ have a large degree of noise in them due to the circuit-privacy call that was used to rerandomize the ciphertexts. Therefore, the first thing that is done is that the ciphertexts are re-encoded with less noise using the same procedure as FHE bootstrapping.) Next, compute ciphertext z_i of the result $f(x_1, \ldots, x_n)$. Note that each player can complete this step using only local information (since the public key for the FHE includes all the information needed for bootstrapping etc).

5: Each player P_i that has not been excluded broadcasts the ciphertext z_i computed in the previous step. Each player then locally computes the majority of the broadcasts as ciphertext z'. A majority is guaranteed to exist since the malicious players form a minority and **Eval** is deterministic. Any player whose broadcast differs from the majority is excluded from the remaining portion of the protocol.

6: Players p_i that have not been excluded run the distributed subprotocol $\tilde{D}(z', [^\star s_1], \ldots, [^\star s_n])$ using input z' and their local share $[^\star s_i]$. The output of the protocol is taken as the output.

Figure 5.16: THE MULTIPARTY COMPUTATION PROTOCOL

real execution of the protocol there exists an ideal-world adversary A', sometimes referred to as a simulator, which "achieves the same effect" in the ideal world. This is made more precise in what

follows.

The real model.

Let π be a multi-party protocol computing a circuit f. We consider an execution of π on an open broadcast network with rushing in the presence of a statically-corrupting adversary $\mathcal{A}$ coordinated by a non-uniform environment $Z = \{Z_k\}$. At the outset of the execution, Z gives I and z to $\mathcal{A}$, where $I \subset [n]$ represents the set of corrupted parties and z denotes an auxiliary input. Then the environment gives input x_i to each party P_i and gives $\{x_i\}_{i\in I}$ to $\mathcal{A}$. The parties then run the protocol π with $\mathcal{A}$ providing the messages sent on behalf of any corrupted party. At the end of the execution, $\mathcal{A}$ gives to Z an output which is an arbitrary function of $\mathcal{A}$'s view thus far, and Z is additionally given the outputs of the honest parties. If the adversary aborts the protocol at some step (formally, if the output of some honest party at the end of the phase is $\perp$), execution is halted; otherwise, execution continues until the protocol is finished. Once the execution terminates, Z outputs a bit; we let $\mathrm{REAL}_{\pi,\mathcal{A},Z}(k)$ be a random variable denoting the value of this bit.

The ideal model.

In the ideal model, there is a trusted party who computes f on behalf of the parties. This definition of ideal model corresponds to a notion of security where fairness and output delivery are guaranteed. Once again, we have an environment Z which provides inputs $x_1, \ldots, x_n$ to the parties, and provides $I, \{x_i\}_{i\in I}$, and z to $\mathcal{A}'$. At the outset, Z gives I and z to $\mathcal{A}'$ and provides input x_i to party P_i and gives $\{x_i\}_{i\in I}$ to $\mathcal{A}'$. Each honest party sends their input to the trusted party; adversary $\mathcal{A}'$ sends inputs on behalf of players in I and can also send the special symbol $\perp$ to the trusted party. The trusted party computes $y \leftarrow f(x_1, \ldots, x_n)$ using the inputs it receives from the players. For each player that submits $\perp$, the trusted party uses input 0. Finally, the trusted party delivers output y to each player who submits an input that is not $\perp$.

At the end of this phase, $\mathcal{A}'$ gives to Z an output which is an arbitrary function of its view thus far, and Z is additionally given the outputs of the honest parties. After all phases have been completed, Z outputs a bit. Once again, we let $\mathrm{IDEAL}_{\pi,\mathcal{A}',Z}(k)$ be a random variable denoting the

value of this bit. With the above in place, we can now define our notions of security.

Definition 5.5.1. *(Security) Let π be a multi-party protocol for computing a circuit f, and fix $s \in \{1, \ldots, n\}$. Then we say that π securely computes f in the presence of malicious adversaries corrupting s parties if for any ppt adversary $\mathcal{A}$ there exists a ppt adversary $\mathcal{A}'$ such that for every polynomial-size circuit family $Z = Z_k$ corrupting at most s parties the following is negligible:*

$$\left|\Pr\left[\mathrm{REAL}_{\pi,\mathcal{A},Z}(k) = 1\right] - \Pr\left[\mathrm{IDEAL}_{f,\mathcal{A}',Z}(k) = 1\right]\right|.$$

The Simulation

We present a simulator $\mathcal{A}'$ for an adversary $\mathcal{A}$ that coordinates a group of corrupted players C. The steps of the simulation are as follows:

1. Begin an execution of the protocol for the adversary $\mathcal{A}(1^k, n, C, \{x_c\}_{c \in C})$ in which the state for the honest players P_i is initialized with input $(1^k, n, x_i = 0)$ and is thereafter maintained internally by the simulator.

2. Run $(\mathrm{PK}, \mathrm{SK}_1, \ldots, \mathrm{SK}_n) \leftarrow \tilde{G}(\eta, \tau, \rho, \theta, \Theta, \kappa)$ described in §5.4.3 to generate a public and secret key for the threshold FHE scheme. Run the simulator $S_{\tilde{G}}(\mathrm{PK} = (\langle \vec{x'} \rangle, \vec{y'}, \vec{c'}), \eta, \tau, \rho, \theta, \Theta, \kappa)$ described in §5.4.3 for the $\tilde{G}$ key generation protocol with input PK using the adversary $\mathcal{A}$ and using the internal copies of the honest players. The states of the adversary $\mathcal{A}$ and the internal copies of the honest player are maintained at the end of the simulation.

3. For each honest player, use a randomly generated ciphertext under the public-key PK corresponding to 0 as its input, and follow the remaining procedure for running the coinflipping and hidden-bits POK protocol honestly. When it is time for a corrupted player $p_c \in C$ to run the POK, if the first execution succeeds, then invoke algorithm *Extractor* for the POK to recover the input x_c. If the first execution fails, then exclude the user from future simulated steps of the protocol. If the extractor fails, then abort the simulation.

4. Feed the inputs $\{x_c\}_{c\in C}$ for the corrupted players to the external trusted party and wait for a response output y.

5. Follow steps 4 and 5 of the Protocol on behalf of the honest players based on the broadcast input ciphertexts. After step 4, each internal copy of the honest player has computed a value z_i that they broadcast. In step 5, each internal honest copy broadcasts z_i; each corrupted player that does not broadcast z_i is excluded from the rest of the internal simulation. Compute z' for these honest players as per the protocol.

6. Run the simulator $S_{\tilde{D}}$ (see §5.4.4) on the input z', the shares of $[^\star s_i]$'s for the malicious parties and the value y to simulate the decryption protocol for $\mathcal{A}$.

7. Output whatever the adversary $\mathcal{A}$ outputs.

Theorem 5.5.2. *Let π be the protocol described in Figure 5.16 for a function f, and fix $s \in \{1,\ldots,n/2\}$. Under the appropriate Approximate-GCD and sparse subset sum assumptions, it holds that for for any ppt adversary $\mathcal{A}$, there exists a ppt adversary $\mathcal{A}'$ such that for every polynomial-size circuit family $Z = Z_k$ corrupting a minority of parties the following is negligible:*

$$\left|\Pr\left[\mathrm{REAL}_{\pi,\mathcal{A},Z}(k)=1\right]-\Pr\left[\mathrm{IDEAL}_{f,\mathcal{A}',Z}(k)=1\right]\right|.$$

Proof. (High level Idea) We present a high-level summary of the changes needed in the security proof from [68]. Our proof summary consists of a series of hybrid experiments that relate REAL and IDEAL and a brief description on why two consecutive hybrid experiments are indistinguishable.

$\mathbf{Hybrid}_1(1^k,\mathcal{A},Z)$**:** This hybrid experiment is the same as the real experiment REAL except that the experiment first generates $(\mathrm{SK},\mathrm{PK}) \leftarrow \tilde{G}(\eta,\tau,\rho,\theta,\Theta,\kappa)$ and then runs the simulator $S_{\tilde{G}}(\mathrm{PK}=(\langle\vec{x'}\rangle,\vec{y'},\vec{c'}))$ interacting with the adversary $\mathcal{A}$.

We claim that REAL and $\mathbf{Hybrid}_1$ are identical because $S_{\tilde{G}}$ is information theoretically-secure.

$\mathbf{Hybrid}_2(1^k, \mathcal{A}, Z)$: This hybrid experiment is the same as the previous one, except that the extractor for the Hidden Bit POK is used on each broadcast ciphertext from the adversary $\mathcal{A}$. If any extraction fails, then the experiment aborts.

We claim that the $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_1$ distributions are statistically close. The only difference occurs when the extraction fails in the second hybrid for one instance of the Hidden Bit POK protocol. By the proof of knowledge extraction error property from the Hidden POK protocol proven in Theorem 5.3.2 and the union bound, it follows that these events occur with a negligibly small probability, and therefore the distributions are statistically close.

$\mathbf{Hybrid}_3(1^k, \mathcal{A}, Z)$: This hybrid experiment is the same as the previous, except that the experiment sends the extracted input values for the malicious parties to the trusted party and receives output $y = f(x_1, \ldots, x_n)$ in return. The experiment then uses the simulator $S_{\tilde{D}}$ (see §5.4.4) for the decryption on input z, the shares of $[^\star s_i]$'s for the malicious parties and the value y to force the players to output y. (Notice that at this point, z corresponds to an encryption of y, but that the simulator is used to feed messages to the adversary instead of the threshold decryption protocol.)

We claim that $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ are computationally indistinguishable by the simulation property for the threshold decryption protocol and the unique decoding property of the POK protocol. In particular, the decoding property in Theorem 5.3.2 states that the inputs extracted from the adversaries will be the same as the inputs decrypted using SK (i.e., the inputs used in the real protocol computation). Thus, conditioned on this event that all inputs are consistent between the two experiments, the value y returned from the trusted party corresponds to the ciphertext z. Finally, the simulation property of $S_{\tilde{D}}$ guarantees that the transcripts between the two hybrids are identical.

$\mathbf{Hybrid}_4(1^k, \mathcal{A}, Z)$: This hybrid experiment is the same as the previous, except that the input 0 is used to produce a ciphertext and run the Hidden Bit POK for each of the honest parties.

We claim that $\mathbf{Hybrid}_4$ and $\mathbf{Hybrid}_3$ are computationally indistinguishable based on the soundness of the hidden POK and the hidden bit property from Theorem 5.3.6. In particular,

suppose that these two distributions were distinguishable with advantage ϵ. We define more hybrid experiments $\mathbf{Hybrid}_{3,i,j}$ in which all of the input bits up until the j^{th}bit of player i are formed using the 0 input, whereas the rest of the input bits use the honest player inputs. Note that $\mathbf{Hybrid}_{3,1,0} = \mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_{3,n,n} = \mathbf{Hybrid}_4$. Thus, there exists a pair of consecutive experiments $\mathbf{Hybrid}_{3,i,j}$ and $\mathbf{Hybrid}_{3,i,j+1}$ (without loss of generality, we assume this boundary occurs between j and $j+1$ instead of across the last bit of one player and the first bit of the next player and that the difference is between 0 and 1 in these positions) with advantage ϵ/n^2. For convenience, we denote this pair of hybrid experiments $\mathbf{Hybrid}_a$ and $\mathbf{Hybrid}_b$. We will now use this pair to violate the soundness of the POK, or the simulation properties.

We first claim that inputs $\{x_c\}^a$ extracted from the parties in C in $\mathbf{Hybrid}_a$ and the ones $\{x_c\}^b$ will be the same with all but negligible probability (If extraction fails, we use $\perp$ to denote the extracted input. As argued earlier, the probability of extracting $\perp$ in $\mathbf{Hybrid}_3$ is negligible). Suppose this is not true: i.e. $\Pr[\mathbf{Hybrid}_b(A)$ extracts $\{x_c\}^b \neq \{x_c\}^a] > \mu(k)$. Then there exists a vector of inputs $x = (x_1, \ldots, x_n)$ for which the probability that these two sets are different is greatest (inverse polynomial probability of success); let this vector, the position j, and the sets $\{x_c\}^a$ and $\{x_c\}^b$ be given as non-uniform advice for the following adversary $\mathcal{A}'$ that breaks the hidden POK property. The adversary $\mathcal{A}'$ runs the hybrid $\mathbf{Hybrid}_a$ using the inputs x while participating in the Hidden POK game. It receives a PK externally from the Hidden POK game and uses this PK with the simulator $S_{\tilde{G}}$ in the first step of $\mathbf{Hybrid}_a$. It then receives a ciphertext under PK from the external game and uses it (along with its decryption opening query in the HB game) to run the (i,j) instance of the input bit protocol in $\mathbf{Hybrid}_a$. Finally, it runs the extractor for all of the malicious parties to recover a set of inputs $I = \{x_c\}_{c \in C}$. If $I = \{x_c\}^a$, then the adversary output 0, and otherwise outputs 1. Notice that if the input ciphertext is 0, then the adversary has run experiment

$\mathbf{Hybrid}_a$, whereas if the input is 1, the adversary runs $\mathbf{Hybrid}_b$ and so:

$$\begin{aligned}\Pr[HB_{\mathcal{A}'}(1^k) = 1] = &\Pr[b = 0] \Pr[\mathcal{A}' \text{ outputs } 0 \mid b = 0] + \Pr[b = 1] \Pr[\mathcal{A}' \text{ outputs } 1 \mid b = 1] \\ &\frac{1}{2} \Pr[\mathbf{Hybrid}_a(\mathcal{A}) \text{ extracts } \{x_c\}^a] + \frac{1}{2} \Pr[\mathbf{Hybrid}_b(\mathcal{A}) \text{ extracts } \{x_c\}^b \neq \{x_c\}^a] \\ &\geq \frac{1}{2} + \mu(k)/2\end{aligned}$$

Thus, our claim that the extracted outputs must be the same w.h.p holds. Conditioned on this event that both extracted sets are equal, it follows that the value y recovered in $\mathbf{Hybrid}_4$ and the decryption of z from $\mathbf{Hybrid}_3$ will be the same. However, in $\mathbf{Hybrid}_4$, the ciphertext z corresponds to a different plaintext, namely $f(\{x_c\}^a, 0, \ldots, 0)$ where 0 is used for the honest players. We now claim that this difference is indistinguishable by introducing another hybrid experiment $\mathbf{Hybrid}_{3z}$ in which the same ciphertext from $\mathbf{Hybrid}_3$ is given to the experiment and used in place of the encryption generated in $\mathbf{Hybrid}_4$. We claim—through a standard argument—that $\mathbf{Hybrid}_{3z}$ and $\mathbf{Hybrid}_4$ must be indistinguishable based on the semantic security of the threshold encryption scheme. The only remaining difference to account for is the use of the simulator $S_{\tilde{D}}$ for decryption. The stand-alone security of this simulator implies our claim. □

Bibliography

[1] Robert Richardson. 2010/2011 computer crime and security survey, 2011. http://gocsi.com/survey.

[2] What data thieves dont want you to know: The facts about encryption and tokenization, 2012. http://tiny.cc/xo3knw.

[3] Larry Dignan. Sony's data breach costs likely to scream higher, 2011. http://tiny.cc/2n3knw.

[4] 2010 annual study: U.S. cost of a data breach, 2010. http://tiny.cc/si3knw.

[5] Barack Obama. Remarks by the president on securing our nation's cyber infrastructure, 2009. http://tiny.cc/534knw.

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

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

[8] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, pages 48–62, 2004.

[9] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[10] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *CRYPTO*, pages 271–289, 2006.

[11] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In *TCC*, pages 427–444, 2008.

[12] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded cca2-secure encryption. In *ASIACRYPT*, pages 502–518, 2007.

[13] Takahiro Matsuda and Kanta Matsuura. Parallel decryption queries in bounded chosen ciphertext attacks. In *Public Key Cryptography*, pages 246–264, 2011.

[14] Yael Gertner, Tal Malkin, and Steven Myers. Towards a separation of semantic and cca security for public key encryption. In *TCC*, pages 434–455, 2007.

[15] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *FOCS*, pages 531–540, 2010.

[16] Tatsuaki Okamoto and David Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA*, pages 159–175, 2001.

[17] Steven Myers and Abhi Shelat. Bit encryption is complete. In *FOCS*, pages 607–616, 2009.

[18] Rafael Pass and Abhi Shelat. A course in cryptography, 2010. http://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf.

[19] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, pages 433–444, 1991.

[20] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO*, pages 26–45, 1998.

[21] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[22] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.

[23] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[24] Leslie Lamport. Constructing digital signatures from one-way functions. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[25] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.

[26] Alexander W. Dent. The hardness of the dhk problem in the generic group model. *IACR Cryptology ePrint Archive*, 2006:156, 2006.

[27] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[28] Steven Myers and Abhi Shelat. Bit encryption is complete. In *FOCS*, pages 607–616, 2009.

[29] Alexander W. Dent. The cramer-shoup encryption scheme is plaintext aware in the standard model. In *EUROCRYPT*, pages 289–307, 2006.

[30] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.

[31] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *CRYPTO*, pages 314–332, 2010.

[32] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[33] Yehuda Lindell. A simpler construction of cca2-secure public-keyencryption under general assumptions. *J. Cryptology*, 19(3):359–377, 2006.

[34] Yael Gertner, Tal Malkin, and Steven Myers. Towards a separation of semantic and cca security for public key encryption. In *TCC*, pages 434–455, 2007.

[35] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.

[36] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In *TCC*, pages 419–436, 2009.

[37] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *EUROCRYPT*, pages 673–692, 2010.

[38] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, 1998.

[39] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *EUROCRYPT*, pages 275–288, 2000.

[40] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

[41] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO*, pages 426–442, 2004.

[42] Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-kem/dem: A new framework for hybrid encryption and a new analysis of kurosawa-desmedt kem. In *EUROCRYPT*, pages 128–146, 2005.

[43] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.

[44] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[45] David Cash, Eike Kiltz, and Victor Shoup. The twin diffie-hellman problem and applications. *J. Cryptology*, 22(4):470–504, 2009.

[46] Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In *EUROCRYPT*, pages 313–332, 2009.

[47] Goichiro Hanaoka and Kaoru Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational diffie-hellman assumption. In *ASIACRYPT*, pages 308–325, 2008.

[48] Kristiyan Haralambiev, Tibor Jager, Eike Kiltz, and Victor Shoup. Simple and efficient public-key encryption from computational diffie-hellman in the standard model. In *Public Key Cryptography*, pages 1–18, 2010.

[49] Yevgeniy Vahlis. Two is a crowd? a black-box separation of one-wayness and security under correlated inputs. In *TCC*, pages 165–182, 2010.

[50] Ronald Cramer, Dennis Hofheinz, and Eike Kiltz. A twist on the naor-yung paradigm and its application to efficient cca-secure encryption from hard search problems. In *TCC*, pages 146–164, 2010.

[51] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989.

[52] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle diffie-hellman assumptions and an analysis of dhies. In *CT-RSA*, pages 143–158, 2001.

[53] Dennis Hofheinz and Eike Kiltz. The group of signed quadratic residues and applications. In *CRYPTO*, pages 637–653, 2009.

[54] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, pages 445–465, 2010.

[55] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[56] Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *STOC*, pages 590–599, 2001.

[57] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[58] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *ASIACRYPT*, pages 70–88, 2011.

[59] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.

[60] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.

[61] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, pages 483–501, 2012.

[62] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. Cloud-assisted multiparty computation from fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:663, 2011.

[63] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, pages 169–188, 2011.

[64] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, pages 420–432, 1991.

[65] Yuval Ishai, Jonathan Katz, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On achieving the "best of both worlds" in secure multiparty computation. *SIAM J. Comput.*, 40(1):122–141, 2011.

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

[67] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO*, pages 120–127, 1987.

[68] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299, 2001.

[69] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT*, pages 162–177, 2000.

[70] Matthew Green and Susan Hohenberger. Cpa and cca-secure encryption systems that are not 2-circular secure. *IACR Cryptology ePrint Archive*, 2010:144, 2010.

[71] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT*, pages 311–326, 1999.

[72] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.

[73] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[74] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *TCC*, pages 285–304, 2006.

[75] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *Public Key Cryptography*, pages 343–360, 2007.

[76] Tomas Toft. Constant-rounds, almost-linear bit-decomposition of secret shared values. In *CT-RSA*, pages 357–371, 2009.

[77] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *PODC*, pages 201–209, 1989.

[78] Eike Kiltz, Gregor Leander, and John Malone-Lee. Secure computation of the mean and related statistics. In *TCC*, pages 283–302, 2005.

[79] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[80] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[81] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[82] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456, 1991.

[83] Jonathan Herzog, Moses Liskov, and Silvio Micali. Plaintext awareness via key registration. In *CRYPTO*, pages 548–564, 2003.

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

[85] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, pages 145–161, 2003.

[86] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, pages 705–714, 2011.

[87] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, pages 52–78, 2007.

[88] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In *TCC*, pages 595–613, 2009.

[89] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

[90] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *CCSW*, pages 113–124, 2011.

[91] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.

[92] Seung Geol Choi, Ariel Elbaz, Tal Malkin, and Moti Yung. Secure multi-party computation minimizing online rounds. In *ASIACRYPT*, pages 268–286, 2009.

[93] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO*, pages 247–264, 2003.

[94] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, pages 501–520, 2006.

[95] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In *CRYPTO*, pages 241–261, 2008.

[96] Moni Naor and Kobbi Nissim. Communication complexity and secure function evaluation. *CoRR*, cs.CR/0109011, 2001.

[97] Zuzana Beerliová-Trubíniová and Martin Hirt. Efficient multi-party computation with dispute control. In *TCC*, pages 305–328, 2006.

[98] Martin Hirt and Jesper Buus Nielsen. Robust multiparty computation with linear communication complexity. In *CRYPTO*, pages 463–482, 2006.

[99] Martin Hirt, Jesper Buus Nielsen, and Bartosz Przydatek. Asynchronous multi-party computation with quadratic communication. In *ICALP (2)*, pages 473–485, 2008.

[100] Matthias Fitzi, Martin Hirt, Thomas Holenstein, and Jürg Wullschleger. Two-threshold broadcast and detectable multi-party computation. In *EUROCRYPT*, pages 51–67, 2003.

[101] Martin Hirt and Ueli M. Maurer. Robustness for free in unconditional multi-party computation. In *CRYPTO*, pages 101–118, 2001.

[102] Martin Hirt, Ueli M. Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In *ASIACRYPT*, pages 143–161, 2000.

[103] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In *CRYPTO*, pages 121–136, 1998.

[104] Martin Hirt, Jesper Buus Nielsen, and Bartosz Przydatek. Cryptographic asynchronous multi-party computation with optimal resilience (extended abstract). In *EUROCRYPT*, pages 322–340, 2005.

[105] Martin Hirt and Jesper Buus Nielsen. Upper bounds on the communication complexity of optimally resilient cryptographic multiparty computation. In *ASIACRYPT*, pages 79–99, 2005.

[106] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, pages 521–536, 2006.

[107] Ronald Cramer, Vanesa Daza, Ignacio Gracia, Jorge Jiménez Urroz, Gregor Leander, Jaume Martí-Farré, and Carles Padró. On codes, matroids and secure multi-party computation from linear secret sharing schemes. In *CRYPTO*, pages 327–343, 2005.

[108] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT*, pages 596–613, 2003.

[109] Ronald Cramer, Ivan Damgård, and Stefan Dziembowski. On the complexity of verifiable secret sharing and multiparty computation. In *STOC*, pages 325–334, 2000.

[110] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, pages 316–334, 2000.

[111] Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In *CRYPTO*, pages 558–576, 2010.

[112] Ivan Damgård and Marcel Keller. Secure multiparty aes. In *Financial Cryptography*, pages 367–374, 2010.

[113] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography*, pages 325–343, 2009.

[114] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *Public Key Cryptography*, pages 160–179, 2009.

[115] Ivan Damgård, Jesper Buus Nielsen, and Daniel Wichs. Universally composable multiparty computation with partially isolated parties. In *TCC*, pages 315–331, 2009.

[116] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In *CRYPTO*, pages 241–261, 2008.

[117] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO*, pages 572–590, 2007.

[118] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, pages 501–520, 2006.

[119] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.

[120] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT*, pages 311–326, 1999.

[121] Tal Rabin. A simplified approach to threshold and proactive rsa. In *CRYPTO*, pages 89–104, 1998.

[122] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Optimal resilience proactive public-key cryptosystems. In *FOCS*, pages 384–393, 1997.

[123] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Proactive RSA. In *CRYPTO*, pages 440–454, 1997.

[124] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO*, pages 307–315, 1989.

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

[126] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.