8-10-2015

Graded Cryptographic Primitives

Murat Osmanoglu
University of Connecticut - Storrs, murat.osmanoglu@uconn.edu

Graded Cryptographic Primitives

Murat Osmanoglu, Ph.D.

University of Connecticut, 2015

ABSTRACT

This thesis studies a particular functionality for privacy-preserving systems, that allows a user to demonstrate a proof showing that the user has been approved by a number of authorities, without revealing their identities. We first consider this functionality for two fundamental cryptosystems: digital signature schemes, and public key encryption schemes, and introduce a new notion "grade" for these systems. Within this scope, we formalize two new primitives, graded signatures and graded encryption.

Graded signature schemes enable a user to consolidate a set of signatures on a message m originating from l different signers. The resulting consolidated signature object on m reveals nothing more than the grade of the signature and the validity of the original signatures without leaking the identity of the signers. On the other hand, graded encryption schemes allow a sender to specify a numerical grade i for the ciphertext during the encryption depending on the importance of the message. Users can only decrypt messages directed to their identity at grade i as long as they have contacted i authorities in sequential order. We present efficient constructions and useful applications such as multi-stage games (e.g., "who wants to be a millionaire") played in a distributed fashion for graded encryption and anonymous petition system

for graded signatures.

In systems having a large number of participants, e.g., large scale privacy-preserving petitions, a graded signature scheme with linear size signatures will not be an efficient tool in practice. We observe that if we distribute the signing keys of the scheme associated to different grades in an efficient way, we can obtain a graded signature scheme that enjoys constant size signatures. In this direction, we revisit the problem of minimizing the share size of a multi-secret sharing scheme (MSSS). To circumvent the information-theoretic lower bound (Blundo [13]), we focus on the computational setting, and present an efficient construction of the MSSS with share size only logarithmic in the number of secrets (hence effectively optimal).

Graded Cryptographic Primitives

Murat Osmanoglu

M.S., Yildiz Technical University, 2007

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2015

APPROVAL PAGE

Doctor of Philosophy Dissertation

Graded Cryptographic Primitives

Presented by

Murat Osmanoglu, M.S. Math.

Co-Major Advisor _____
Aggelos Kiayias

Co-Major Advisor _____
Alexander Russell

Associate Advisor _____
Donald Sheehy

Associate Advisor _____
Marten Van Dijk

University of Connecticut

2015

ACKNOWLEDGMENTS

Praise and glory to Allah Who
brings down the wisdom into the
heart of the words

Ibn Arabi

First of all, I would like to express my exceptional appreciation to my advisor Dr. Aggelos Kiayias for his contributions to my academic and personal growth. Not only did he unceasingly hearten me in my research but also he taught me the essential skills of growing as a research scientist. Besides it was always fun to discuss my studies with him. He was always patient towards me and never refrained from offering his full support whenever necessary. In December 2013 I was his guest in Greece for a few weeks, which gave me the chance to witness his sincere hospitality as well. I am deeply grateful. And my deep gratitude also goes to my co-major advisor Dr. Alexander Russell, whose guidance and ever-present support have always illuminated my path. Our endless discussions were always fruitful and his recommendations made a major impact in getting the results in this thesis. It was a pleasure to get to know and to work with them both.

My appreciation also extends to Dr. Donald Sheehy Dr. Marten Van Dijk members of my thesis committee. Their mentoring and helpful recommendations made the thesis-writing process a lot easier for me, and they were kind enough to take the time

and attend my committee. I should also thank specifically to Dr. Bulent Yener, who accepted to become an external member of my committee despite his busy schedule.

It is of utmost importance that I acknowledge the contributions of my friend and colleague Qiang. I could not have managed to write this thesis (or even to complete my studies) if it was not for his support and companionship. Never has he withheld his hand in times necessary and has always encouraged me for the better. Shared experiences like our travels and stays for conferences showed me what an amazing personality he has outside the lab as well. I want to thank him deeply for his generosity and good fellowship; it was a privilege to work with you my dear friend! Besides I would like to thank my friend and lab-mate Ozgur for his assistance and support.

I also had the opportunity to form very precious friendships throughout my studies in CT, which made long hours of stressful study a lot more tolerable. To begin with, I should thank my roommate Ahmet for his tolerance and patience towards me. I must thank Sadullah, Said, Cemil and Turgut for their company and their assistance in reducing the stress of my studies, and to Kadir, specifically for his remote navigational assistance! Most of all I want to thank Hakan for never losing his faith in me and for knowing how to motivate me when I needed it most.

I am indebted to my mother for her presence and for her endless prayers, my father for never giving up on me, and my siblings for always being there for me. I could not have accomplished this without their faith.

Last but not the least, I should thank Turkish Government for giving me the opportunity to study in the US and to the staff of the New York Offices of the Turkish Educational Attach for dealing with my educational problems and solving them on time.

Contents

Chapter 1

Introduction

A petition is a formal demand that enables a number of people to submit their requests about a certain issue to the relevant authority. In a regular (paper-based) petition system, each individual provides a handwritten signature together with a unique identifier in order to allow the authentication of the petition. Providing the signatures with the identifiers also enables the receiver to detect the duplicates or forged signatures effectively. However, collecting and verifying signatures by hand requires a lot of time. Alternatively, the digital petition presents remarkable advantages: (i) it allows the petitioner to reach out more people, and (ii) it enables the verification process to be automated, and to be implemented efficiently. But, the traditional digital petition system also presents some challenge.

Consider the public service that White House provides, called We the People, which enables the citizens to create and sign petitions for certain issues in order to motivate the federal government to act on them. There are two types of thresholds in the system: (i) the number of the signatures should exceed 150 to be searchable

within the web site, (ii) the number of the signatures should exceed 100000 to require response. If a petition crosses the second threshold within the designated period, the White House will respond. The system requires the users need to provide some information that can be considered as personal sensitive data [1], in order to register system and sign a petition. However, citizens may not want to reveal their personal information to the government when their signatures are demanded on a petition, especially about sensitive issues such as political opinion or religious belief. Thus, this feature of the system may cause some loss of support in those particular issues. Alternatively, a petition system that enables the signers to remain anonymous would be desired in this sense.

Let us clarify it with another example. In many online communities, reputation has become an essential instrument for evaluating trust. It can be viewed as a component of identity as created by other's opinions. Consider the internet company "Yelp", a social-networking site, that provides an environment for users to submit reviews on the products or services of local businesses such as restaurants. These reviews enable each business to develop its reputation.

In all such systems, reputation affects the pseudonyms of the users rather than the users, but the pseudonyms can be used to associate the users with a history of their activities. However, in most environments, anonymity is a desired feature for users, i.e., they do not want the pseudonyms to be used to identify their history. Consider a message board similar to one discussed in [9]. The writers share some posts in the board, and the readers rate the posts depending on their accuracy. Then, the writers create their reputation based on the ratings collected from the readers, and attach this

[1] According to Art. 8 of EU Protection Directive, the processing of personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs shall be prohibited by the member states.

reputation to new post to be shared in order to increase the the attention the posts may get. The readers now can evaluate these posts based on the writers' reputation. Thus, this message board enables us to highlight valuable posts. However, we want the board to provide anonymity to the readers when they rate posts. Similar to the privacy-preserving petition system discussed above, we want a mechanism that enables one to present the approvals collected from distinct entities without revealing their identities. In other words, we are interested in a special functionality in a privacy-preserving setting, that allows a user to reveal only the number of sources that he has contacted.

Within this framework, we introduce a new primitive that we call "graded signatures", which meets the requirements discussed above. The primitive enables a user to present a consolidation of a number of signatures (say ℓ) collected from ℓ distinct signers on the same message without revealing the identities of signers. We call ℓ, "grade" of the combined signature. If there are n registered signers, the grade will range in $[1, n]$. The grade does not need to be determined before the signature collecting procedure. In other words, the scheme allows the user to recombine the new signatures collected from new signers to the graded signature he holds, and to produce a graded signature on the same message with higher grade. In the chapter 5, we give the formal definition of the primitive and discuss the security requirements: unforgeability and anonymity. We carefully compare our graded signature with the existing related primitives. Also, we present an efficient construction that achieves constant size secret key and public key. Furthermore, we manage to obtain linear size graded signatures using recent results in efficient range proofs [26].

We consider the notion of "grade" as a general concept, and search its feasibility for other primitives. Within this concept, we study the primitive "graded encryp-

tion" as an extension of identity-based encryption to the graded settings. In graded encryption, there will be a number of authorities corresponding to fixed indices. Each authority here holds a master secret key to be used to issue a partial key to the users. The primitive allows one to set a grade for a ciphertext in the encryption process according to the significance of the message. The receiver of the ciphertext can only decrypt it if he possess a secret key with same grade. The grade here determines the number of different partial keys (= the number of distinct authorities) the receiver must collect from the authorities in sequential order. We also emphasize some security requirements for graded encryption such that a secret key with a certain grade would not be helpful in decrypting a ciphertext with higher grade. In this sense, the primitive guarantees that the content of a ciphertext with grade i will be secure even if all authorities up to i except one have been corrupted. Also, we present an efficient construction for graded encryption which enjoys constant-size secret keys and ciphertexts. We also extend it to a two-mode version which is suitable for the application of the game "who wants to be a millionaire?". In a two-mode graded encryption scheme, there are two types of secret keys: type-1 secret key is used to decrypt a ciphertext with same level, and cannot be upgraded; on the contrary, type-2 secret key can be upgraded, but cannot be used to decrypt a ciphertext with same level. We also promote the primitive with useful applications that we discuss in chapter 6

For systems with a large number of participants such as large scale privacy-preserving petitions, a graded signature scheme with linear-size signatures will not be an efficient tool in practice. Instead, a scheme in which the signature size is independent of the number of signers would be better. However, obtaining such scheme is a challenging task. A trivial way could be to run an (i, n)-threshold signature scheme

for each signing key of level i [2]. Although we achieve constant size signatures in this trivial solution, the size of the secret key that each signer holds will be linear in the number of levels since a constant size secret share is given to each signer for the signing key of each level. It's not practical since it requires each signer to hold a linear-size secret key. However, if we find an efficient way to share n signing keys among the signers, we could get a practical graded signature scheme by utilizing a threshold signature scheme. Thus, we now consider the problem of distributing n secrets $s_1, ..., s_n$ among n parties in a way that; (i) each secret s_i can be constructed from at least i corresponding shares, and (ii) the size of the share that each party holds is much smaller than linear-size. This can be viewed as a variant of multi-secret sharing scheme, and we call it "graded secret sharing".

In a multi-secret sharing scheme (MSSS), n secrets $s_1, ..., s_n$ with the threshold values $t_1, ..., t_n$ are distributed among parties in a way that any t_i parties can recover the corresponding secret s_i, but fewer than t_i parties get no information about it. Blundo et al. [13] showed that in order to achieve unconditional security in MSSS, the size of the share that each party holds must be at least linear in the number of secrets. We cannot hope better than this for unconditional security. Thus, we turn our attention to computationally secure MSSS. Krawczyk [66] presented a method to obtain more efficient (t, n)-threshold secret sharing in computational settings. Briefly, the secret is encrypted using a much smaller secret key before the distribution. The ciphertext is shared among parties using an information dispersal scheme [78], and the secret key is distributed using Shamir secret sharing. If the secret key is λ bits long, then the final share size for each user will be $m/t + \lambda$ where m is the size of the

[2] In a (k, n)-threshold signature scheme, the signing key is distributed among n signers in a way that at least k of them should participate to form a signature, i.e. less then k signers can not create a valid signature.

ciphertext.

Krawczyk's construction can easily be adapted to multi-secret sharing by running it n times. Each secret is encrypted under a different secret key. Then the ciphertexts will be distributed using an information dispersal scheme, and the secret keys will be shared using Shamir secret sharing. Thus, the size of each share becomes $\sum_{i=1}^{n} m/t_i + n\lambda$. As we discussed above, our goal here is to obtain smaller size for the share of each party. If each user had only a single secret key, this might reduce the overhead. So, we apply a special type of threshold encryption scheme, called dynamic threshold public key encryption (DTPKE). However, obtaining a DTPKE scheme to achieve our goal is also quite challenging. The only existing construction that enjoys constant-size ciphertext was proposed by Delerablee et al. [33]. The scheme contains $O(n)$ public group elements that are called combining keys, and all elements are required in decrypting even a ciphertext with the threshold 1. Since all combining keys must be given to each party, their scheme is not useful to get an efficient MSSS. Also, requiring the combining keys seems to be inherent in the construction of DTPKE with small size ciphertexts. To this aim, we propose a new DTPKE scheme with a threshold range given in the public parameters. Our new construction uses a degree reduction technique on the exponent and results in a combining key that scales linearly only in the size of the given threshold range rather than the number of parties.

However, even if we use our new construction to apply Krawczyk's idea, it will yield a linear size share for each party. Instead of using it directly for all threshold values in the range $[1, n]$, we first split the range into $\log n$ parts. We then run an independent DTPKE together with an information dispersal scheme for each part in order to distribute the secrets that the associated thresholds fall in the corresponding part. Thus, we get an efficient MSSS that achieves $O(\log n)$ size share for each party.

We also show an interesting application of our MSSS in chapter 4.

Thesis Outline : In Chapter 2, we present notations and some cryptographic definitions and review some basic cryptographic primitives that will be useful in the following chapters. Chapter 3 presents a new flexible construction of dynamic threshold public key encryption scheme (DTPKE) that enables us to obtain an efficient multi-secret sharing scheme (MSSS). In Chapter 4, we first give a generic construction of MSSS from DTPKE, then we present an efficient MSSS using the new construction of DTPKE. In Chapter 5, we propose a new primitive that we call "graded signatures", and present an efficient construction for the primitive. Chapter 6 introduces a new primitive, "graded encryption" together with the useful applications of the new primitive. Finally, in Chapter 7, we give the conclusion of the thesis.

Chapter 2

Preliminaries

In this chapter, we cover the notations and definitions to be used throughout the thesis. We also review some cryptographic tools that will be helpful to comprehend the following chapters. Readers that are familiar with the literature may skip this chapter.

Notations. In the thesis, λ is used as the security parameter that specifies the security level of the cryptographic systems. 1^λ denotes the string of 1's with length λ. The abbreviation "PPT" is denoted probabilistic polynomial time algorithms. An algorithm is called probabilistic polynomial time if it uses randomness and its running time is bounded by some polynomial in the input size. Through the thesis, we use the notion "negligible function" to evaluate the probability that an adversary can break the security of a cryptographic scheme. The negligible function f is a function, that for every possible integer c there exists an integer n_0 such that for all $x > n_0$ $|f(x)| < 1/x^c$. We use the special symbol "$\perp$" as a possible output to indicate the input is invalid. For a given set S, the symbol $u \leftarrow S$ indicates that the element u is

randomly chosen from the set S.

2.1 Bilinear Map

A bilinear map is a pairing that associates a pair of elements from the groups G_1, G_2 with the element in the target group G_T. Bilinear maps are useful tool to build new cryptographic protocols (essentially for pairing-based cryptography), and we'll efficiently utilize them to obtain various new primitive through the thesis.

Let G_1, G_2, and G_T be cyclic groups of prime order p. Let g and $\bar{g}$ be the generators of G_1, G_2, respectively. We say a map $e : G_1 \times G_2 \to G_T$ is a bilinear map if it satisfies the following properties:

- *Bilinearity.* For all $a, b \in Z_p$ and $g_1 \in G_1$, $g_2 \in G_2$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$,
- *Non-degeneracy* $e(g, \bar{g}) \neq 1$,
- *Efficiency* There exists an efficient algorithm [44] to compute $e(g, h)$ for any $g \in G_1$ and $h \in G_2$.

The bilinear pairing e for the bilinear group (p, G_1, G_2, G_T, e) can be classified into three types. If $G_1 = G_2$, the pairing e is classified as type 1. If $G_1 \neq G_2$ and there is an efficiently computable homomorphism $\phi : G_2 \to G_1$, the pairing is classified as type 2. As the last one, if $G_1 \neq G_2$ and there is no efficiently computable homomorphism between these groups, e is classified as type 3.

2.2 Complexity Assumptions

The security of the cryptographic primitives relies on the hardness of some cryptographic problems. We here review some of them that will be used through the thesis.

Discrete Logarithm Problem. The problem was first proposed by Diffie and Hellman [36] as hardness assumption for the cryptographic primitives. Since then, many cryptosystems have been proposed whose security relies on this assumption. The best known general purpose algorithm that solves the discrete logarithm problem has sub-exponential running time in the security parameter.

Let G be a group of order q and g be a randomly chosen generator of G. The discrete logarithm problem is defined as,

$$\text{given } g, h \in G, \text{ find an integer } x \text{ such that } h = g^x.$$

We say that (t, ϵ)-discrete logarithm assumption holds in G if no t-time adversary has advantage at least ϵ in solving discrete logarithm problem.

Diffie-Hellman Problem. The problem was first presented by Diffie and Hellman [36]. It's still believed that computational Diffie-Hellman problem is as hard as the discrete logarithm problem, but it's still an open problem whether hardness of both problems are equivalent.

Let G be a group of order q and g be a randomly chosen generator of G. The computational Diffie-Hellman problem is defined as,

$$\text{given } g, g^a, g^b \in G, \text{ output } g^{ab} \in G.$$

We say that (t, ϵ)-CDH assumption holds in G if no t-time adversary has advantage at least ϵ in solving computational Diffe-Hellman problem in G.

Bilinear Diffie-Hellman Problem. Let G, G_T be two cyclic groups of order q, g be a randomly chosen generator of G. Let $e : G \times G \to G_T$ be a bilinear map as defined in the section 2.1. The computational Bilinear Diffie-Hellman (BDH) problem is defined as,

$$\text{given } g, g^a, g^b, g^c \in G, \text{ output } e(g,g)^{abc} \in G_T.$$

Similarly, the decisional Bilinear Diffie-Hellman (BDH) problem is defined as,

given $g, g^a, g^b, g^c \in G$, and $Z \in G_T$, decide wether $Z = e(g,g)^{abc} \in G_T$ or random in G_T.

We say that (t, ϵ)-BDH assumption holds in G if no t-time adversary has advantage at least ϵ in solving computational Bilinear DIffe-Hellman problem in G. Also, we say that (t, ϵ)-DBDH assumption holds in G if no t-time adversary has advantage at least ϵ in solving decisional Bilinear Diffe-Hellman problem in G.

The Multi-Sequence of Exponents Diffie-Hellman Problem. We here give the cryptographic assumption that the security of our new construction for the dynamic threshold public key encryption is based on. This assumption is essentially the same as the assumption used in [33].

Let $\mathcal{B} = (p, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e)$ be a bilinear group systems, and ℓ, n, and t be three integers. Let g_0 be a generator of $\mathcal{G}_1$ and h_0 be a generator of $\mathcal{G}_2$. Given two random coprime polynomials f and g, of respective orders ℓ and t, with the roots $y_1, \ldots, y_\ell$

and $x_1, \ldots, x_t$ respectively, where $x_i \neq y_j$, and several sequence of exponentiations

$$g_0, g_0^{\gamma}, \ldots, g_0^{\gamma^{\ell+n-1}}, \qquad g^{k.\gamma.f(\gamma)},$$
$$g_0^{\alpha}, g_0^{\alpha.\gamma}, \ldots, g^{\alpha.\gamma^{\ell+n-1}},$$
$$h, h^{\gamma}, \ldots, h^{\gamma^{n-1}},$$
$$h^{\alpha}, h^{\alpha.\gamma}, \ldots, h^{\alpha.\gamma^{n-1}}, \qquad h^{k.g(\gamma)}$$

and also $T \in \mathcal{G}_T$, decide whether T is equal to $e(g,h)^{k.f(\gamma)}$ or some random element of $\mathcal{G}_T$.

Delerablee et al. [33] justified the intractability of the assumption (ℓ, n, t)-MSE-DDH in the generic group model. As they point out, the assumption is non-interactive and easily falsifiable.

2.3 Cryptographic Primitives

2.3.1 Commitment Schemes

A commitment scheme [43, 77] is a two phase-protocol that enables one to commit to a value while keeping it as secret, and to reveal the committed value on demand. The scheme is formed in a way that one cannot change the committed value after committing to it (binding property), and the committed value is not revealed to the receiver of the commitment (hiding property). Formally, a commitment scheme is defined with three algorithms:

- **Setup.** The algorithm takes the security parameter λ and outputs the commitment key CK.

- **Commit.** The algorithm Com takes the value m to be committed and the commitment key CK as inputs. It first picks a random value r as opening and computes the commitment $c = Com(CK, m, r)$. It then outputs the commitment c together with the opening value r.

- **Opening.** The algorithm $Open$ takes the commitment c, the opening value d, and the commitment key CK as inputs, and outputs the value m or $\perp$.

A commitment scheme should satisfy two properties: the commitment gives no information about the value to the receiver (hiding), and the sender cannot open the commitment in two different ways (biding). Formally, these two features are defined in the following way:

- $(t - \epsilon)$-Hiding: For every two values m, m', the distributions $Com(CK, m, r)$ and $Com(CK, m', r)$ are (t, ϵ)-indistinguishable. In other words, with every algorithm A of complexity less than t,

$$|Pr[A(Com(CK, m, r)) = 1] - Pr[A(Com(CK, m', r)) = 1]| \leq \epsilon$$

- Perfectly Binding: For every value m and every two openings r and r',

$$Open(CK, Com(CK, m, r), r') = \perp.$$

2.3.2 Zero-Knowledge Protocols

A zero-knowledge proof [39, 49] allows one party (prover) to convince another party (verifier) that a given statement is true without revealing any information other than this fact. If the prover (or verifier) follows the protocol properly, we call it 'honest'; otherwise we call it 'cheating'. A zero-knowledge proof must satisfy three properties: (1) if the statement is true, the honest verifier will be convinced by an honest prover (completeness), (2) if the statement is false, no cheating prover can convince the honest verifier that it is true (soundness), (3) if the statement is true, the cheating verifier learns nothing other than this fact (zero-knowledge). In this thesis, we deal with the non-interactive variant of zero-knowledge protocols where the interaction is not needed between prover and verifier. Note that Blum et al. [12] showed sharing a common reference string among the parties is enough to achieve non-interactive zero-knowledge.

Let $R = \{(x, w)\}$ be an efficiently computational binary relation, where we call x the statement and w the witness. Let L be the language which consists of the statements from R. A non-interactive argument for a relation R consists of a key generation algorithm G, which creates a common reference string crs, a prover P and a verifier V. The prover generates a non-interactive argument π for an input (crs, x, w). The verifier outputs 1 if the proof is valid; otherwise, outputs 0. Suppose ϵ_1, ϵ_2 are negligible functions,

– A non-interactive argument (G, P, V) is perfectly complete if:

$$\Pr[crs \leftarrow G, \forall (x, w) \in R, V(crs, x, P(crs, x, w))] = 1.$$

– We say (G, P, V) is sound, if $\forall \mathcal{A}$,

$$Pr[crs \leftarrow G; (x, \pi) \leftarrow \mathcal{A}(crs), x \notin L \wedge V(crs, x, \pi) = 1] \leq \epsilon.$$

– (G, P, V) is zero knowledge, if there exists a simulator (S_1, S_2) such that for all non-uniform ppt adversaries $\mathcal{A}$, $\forall (x, w) \in R$

$$|\Pr[crs \leftarrow G, \mathcal{A}^{P(crs,x,w)}(crs) = 1] - Pr[(crs, t) \leftarrow S_1, \mathcal{A}^{S_2(crs,t,x)}(crs) = 1]| < \epsilon$$

.

2.3.3 Range Proofs

Range proofs are zero-knowledge protocols that enable one to convince a verifier that a committed value lies in a specific range. Range proofs are very useful tool in designing various cryptographic schemes such as e-cash, electronic voting, and anonymous credentials. Since Brickell et al. [19] presented the first efficient protocol for the range proofs, there have been a lot of schemes [23, 26, 51, 67] proposed along this line.

Boudot [19] utilized a well known properties of integers in order to get an efficient range proof, i.e. every non-negative integers can be represented as sum of squares. Lipmaa [67] later improved the method to obtain more efficient proofs such that its complexity is independent of the secret size. However, it requires a group with unknown order. Thus, this method is mostly useful when dealing with big secrets.

Bellare and Goldwasser [6] used the idea of the binary decomposition of the secret in order to obtain a range proof for the interval $[0, 2^{\ell}]$. Lipmaa et al. [68] later

gave a general solution for the ranges in the form $[0, b]$. Besides, Chaabouni et al. [26] proposed a non-interactive zero-knowledge range proof using NIZK argument presented by [52], which claimed that a range proof for $a \in [0, H]$ can be constructed if $a = \sum_{i=1}^{n} G_i b_i$ for some G_i and $b_i \in [0, u-1]$ where $(u-1)a \in (u-1).[0, H]$. In the chapter 5, we show how to use the non-interactive range proof given by [26] in order to obtain an efficient graded signature scheme.

2.3.4 Digital Signatures

A digital signature is an authentication mechanism that enables a user to convince a receiver for a message sent by the user, that the message indeed originated from the user. In a digital signature scheme, every user holds a secret key-private key pair. A user creates a signature on a message of his choice using his secret key, and anyone having the user's public key can verify that the signature was generated by the user. Similar to hand-written signatures, digital signatures present two properties; authenticity (a signature convinces a verifier that it was indeed generated by the owner of the public key) and integrity (the signature wasn't modified during transition).

The notion was first introduced by Diffie and Hellman in the seminal paper [36]. Later, Rivest, Shamir and Adleman [79] proposed the first digital signature scheme whose security is based on RSA assumption. Some other works in this line were also proposed by [30,70,75,82]. In 1988, Goldwasser, Micali and Rivest [50] introduced the first rigorous formalization of the security of digital signatures. They also presented the first signature scheme that is secure against an adaptive chosen-message attack.

Formally, a digital signature scheme is defined with three algorithms:

- Setup(λ) : The algorithm takes the security parameter λ as input, and outputs

the secret key sk and the public key pk.

- $\text{Sign}(m, sk)$: The algorithm takes a message m and the secret key sk as inputs, and outputs a signature σ on m.

- $\text{Verify}(m, pk, \sigma)$: The algorithm takes a signature σ, a message m, and the public key pk as inputs. It either outputs accept or reject.

The correctness of the scheme requires that for a given public key pk and the corresponding secret key sk, a signature σ on a message m output by $\text{Sign}(m, sk)$ will always be accepted by the verification algorithm. Formally, $\forall m,$ and λ,

$$Pr[(pk, sk) \leftarrow \text{Setup}(\lambda); \sigma \leftarrow \text{Sign}(m, sk) : \text{reject} \leftarrow \text{Verify}(m, pk, \sigma)] = 0$$

We here consider the standard security definition of digital signature scheme [50]. Intuitively, even if we allow an adversary to make signature queries on messages of his choice, he still cannot forge a valid signature on a message not queried before. Formally, existential unforgeability under chosen message attack is defined with the following game:

- The challenger simulates the **Setup** algorithm and gets the parameters (pk, sk). It then gives the public key mpk to the adversary.

- $\mathcal{A}$ is allowed to make a number of signature queries for the messages of his choice. To respond the queries, the challenger simply runs the Sign algorithm on those messages, and gives the corresponding signatures to the adversary.

- Finally, $\mathcal{A}$ outputs a signature σ^* together with a message m^*, and wins the game if the message m^* was not queried before and $\text{Verify}(m^*, pk, \sigma^*) = \text{accept}$.

The advantage of $\mathcal{A}$ is denoted with $Adv_{\mathcal{A}}^{ef-cma}$. We say a signature scheme is (t, q, ϵ)-existentially unforgeable under adaptive chosen message attack if for all t-time adversaries $\mathcal{A}$ making at most q signature queries, $Adv_{\mathcal{A}}^{cpa}$ is at most ϵ.

2.3.5 Threshold Encryption Schemes

The notion was proposed by Desmedt and Frankel [35] as an extension of public key encryption scheme. In a threshold encryption scheme, the secret key is shared among n decryption servers so that at least t of them should present to correctly decrypt any given ciphertext. In a (t, n)-threshold encryption scheme, there is an entity, called the combiner, that wants to decrypt a ciphertext. When the combiner receives a ciphertext C, he sends it to the decryption servers. The decryption servers then use their secret key share to create the decryption shares, and give them back to the combiner. If the combiner has collected at least t decryption shares, then he combines them to get the plaintext. The security of the scheme guarantees that less than t decryption shares does not reveal anything about the plaintext. We call threshold encryption non-interactive [15, 91] if there is no interaction between the decryption servers in the deception process, and we call threshold encryption scheme robust [41, 46] if the combiner can identify when the decryption servers have given invalid decryption shares. Both of them are desirable features for threshold encryption schemes.

Formally , a threshold public key encryption scheme is defined as follows:

- Setup(t, n, λ) : The algorithm takes the number of servers n, the threshold value t, and the security parameter λ as inputs. It outputs the public key pk and a set of n secret key shares $\{sk_1, ..., sk_n\}$ where each sk_i is associated to the server

i.

- Encrypt(pk, m) : The algorithm takes the public key and a message m as inputs, and outputs a ciphertext c.

- ShareDecrypt(pk, sk_i, c) : The algorithm takes the public key, a ciphertext c, and a secret key share sk_i as inputs. It outputs a decryption share σ_i or $\perp$.

- Combine$(pk, c, \{\sigma_1, ..., \sigma_t\})$: The algorithm takes the public key, a ciphertext c, and a set of decryption shares $\{\sigma_1, ..., \sigma_t\}$ as inputs. It outputs a message m or $\perp$.

If $c =$ Encrypt(pk, m) and $S = \{\sigma_1, ..., \sigma_t\}$ is the set of t distinct decryption shares where $\sigma_i =$ ShareDecrypt(pk, sk_i, c), then the correctness of the scheme requires that Combine$(pk, c, S) = m$.

There is one more algorithm for the robust threshold encryption schemes: ShareVerify, basically checks the validity of the decryption shares using the public key. For such schemes, we also require that for any ciphertext c, if $\sigma =$ ShareDecrpt(pk, sk_i, c), then ShareVerify$(pk, c, \sigma) =$ valid.

The intuition for the semantic security is that a bounded adversary who has corrupted less than t decryption servers cannot get any information about the plaintext. Formally, the security of the scheme is defined with the following game between an adversary and a challenger:

- The adversary $\mathcal{A}$ submits a threshold t.

- The challenger simulates the **Setup** algorithm and gets $(pk, \{sk_1, ..., sk_n\})$. It then gives the public key pk to the adversary.

- $\mathcal{A}$ is allowed to make corrupt queries. To respond the queries, the challenger simply gives the corresponding secret key share sk_i to the adversary.
- $\mathcal{A}$ submits two messages m_0 and m_1 with the threshold t. The challenger randomly choose $b \in \{0, 1\}$, and sends c_b as an encryption of m_b to $\mathcal{A}$.
- $\mathcal{A}$ can adaptively make extra q' corrupt queries.
- Finally, $\mathcal{A}$ outputs a guess b' and wins the game if $b' = b$.

We require in the above game that $q + q' < t$. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{cpa} = |\Pr[b' = b] - \frac{1}{2}|$. We say a threshold public key encryption scheme is semantically secure if for any polynomial time adversary $\mathcal{A}$ the advantage $Adv_{\mathcal{A}}^{cpa}$ is a negligible function over λ.

2.3.6 Identity Based Encryption

Identity-based encryption, proposed by [85], enables one to encrypt a message using the receiver's identity, without the need of the public key and the corresponding certificate of the receiver. In IBE, there is a trusted authority that holds a master secret key, and uses it to issue secret keys to the users based on their identities.

The primitive was introduced by Shamir [85] in 1984, however, the first construction for the notion could be achieved after nearly twenty years independently by Boneh and Franklin [16] and Cocks [29]. The first proposed a scheme based on elliptic curve pairings, and the later proposed a scheme based on the quadratic residuosity problem.

Formally, an identity-based encryption is defined as follows:

- Setup(λ) : The algorithm takes the security parameter λ as input, and outputs the master public key mpk and the master secret key msk for the private key generator (PKG).

- Extract(msk, id) : The algorithm takes the master secret key msk and the identity id of the user as inputs, and outputs the secret key sk_{id} for the user.

- Encrypt(mpk, id, m) : The algorithm takes the master public key mpk, the identity id of the receiver, and a message m as inputs. It outputs a ciphertext c.

- Decrypt(c, sk_{id}) : The algorithm takes a ciphertext and the secret key sk_{id} as inputs, and outputs a message m or $\perp$.

The correctness of the scheme requires that $\forall m$: Decrypt(sk_{id}, c) where $c =$ Encrypt(mpk, id, m) and $sk_{id} =$ Extract(msk, id).

We here consider the semantic security against passive adversaries for identity-based encryption schemes, which was first presented by Boneh and Franklin [16]. The security of the scheme is defined with the following game between an adversary and a challenger:

- The challenger simulates the **Setup** algorithm and gets the parameters (mpk, msk). It then gives the master public key mpk to the adversary.

- $\mathcal{A}$ is allowed to make a number of secret key queries for the identities of his choice. To respond the queries, the challenger simply runs the Extract algorithm on those identities, and gives the corresponding secret keys to the adversary.

- $\mathcal{A}$ submits two messages m_0 and m_1 together with the challenge identity id^*. Note that the adversary is not allowed to choose an identity for which he has already requested a secret key, as challenge identity. The challenger then randomly choose $b \in \{0, 1\}$, and sends c_b as an encryption of m_b to $\mathcal{A}$.

- $\mathcal{A}$ can adaptively make extra secret key queries.

- Finally, $\mathcal{A}$ outputs a guess b' and wins the game if $b' = b$.

The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{cpa} = |\Pr[b' = b] - \frac{1}{2}|$. We say an identity-based encryption scheme is (t, q, ϵ)-semantically secure if for all t-time adversaries $\mathcal{A}$ making at most q secret key queries, $Adv_{\mathcal{A}}^{cpa}$ is at most ϵ.

We here briefly present the construction of the identity-based encryption scheme given by Waters [90]. This construction is used as building block to obtain an efficient graded encryption scheme that will be explained in the chapter 6. Waters proved that the scheme is secure under the decisional BDH assumption. We leave the details to [90].

Let G and G_T be group of prime order p, g be the generator of G, and $e : G \times G \rightarrow G_T$ be a bilinear map. The scheme is defined as follows:

- Setup(λ) : The algorithm choose a random integer $\alpha \in Z_p$. It also chooses random group elements $g_2, u', u_1, ..., u_n \in G$. It sets the value $g_1 = g^{\alpha}$ and outputs the public key as $(g, g_1, g_2, u', \{u_i\})$ and the master secret key as g_2^{α} for the private key generator (PKG).

- Extract(msk, id) : Let id be an n-bit string and I be the set of all i such that $id_i = 1$. PKG first chooses a random integer $r \in Z_p$, then outputs the secret

key of id as

$$sk_{id} = (g_2^{\alpha}.(u' \prod_{i \in I} u_i)^r, g^r).$$

- Encrypt(mpk, id, m) : The algorithm first chooses a random value $t \in Z_p$, then constructs the ciphertext for a message M as follows,

$$C = (e(g_1, g_2)^t.M^t, g^t, (u' \prod_{i \in I} u_i)^t).$$

- Decrypt(c, sk_{id}) : The ciphertext $C = (C_1, C_2, C_3)$ can be decrypted by $sk_{id} = (sk_1, sk_2)$ as

$$C_1.\frac{e(d_2, C_3)}{e(d_1, C_2)}.$$

Chapter 3

Dynamic Threshold Public Key Encryption

In a (t,n)-threshold public key encryption scheme, in order to remove the single point of failure, the decryption key is distributed among a set of n users in such a way that at least t of them should be present to successfully decrypt a ciphertext. In such schemes, the encrypted message will still be secure even if less than t users are corrupted. Threshold public key encryption schemes have extensively been studied [15,24,32,35,60] and many applications such as electronic voting, electronic auctions, key escrow, etc. have been considered in this context.

Regular threshold public key encryption schemes present some limitations: (i) the authorized set is often determined during setup, and (ii) the threshold value t must be fixed at the setup as a part of the public key, and cannot be changed at the encryption time. On the other hand, in some applications the sender of the message may demand some additional flexibility, i.e. for each ciphertext, setting different threshold value or different set of receivers. With this motivation, Ghodosi et al. [60] formalize a

new primitive, called Dynamic Threshold Cryptosystem, that enable the sender to dynamically choose the threshold value t and the authorized set for each ciphertext. They proposed a scheme based on RSA in which the length of the ciphertext is $O(n)$ where n is the number of receivers.

This problem also can be considered as an extension of broadcast encryption to the threshold settings. Within this context, Chai et al. [27] presented an identity-based broadcast threshold decryption scheme in mobile ad hoc network. The scheme enables a node to broadcast an encrypted message to dynamic groups so that only the groups with enough nodes (more than threshold values) can get the message. The scheme also enables the sender to set different threshold values for each group in the target set; however, the length of the ciphertext is still linear in the number of receiver nodes. Daza et al. [32] proposed two threshold encryption schemes (one for PKI setting and one for identity-based setting) that achieves the ciphertext of size $n - t - O(1)$. They utilized some secret sharing techniques and the Canetti-Halevi-Katz transformation [25] to achieve chosen ciphertext security and to shorten the ciphertext size. Intuitively, the sender creates a set of $(n-t)$ dummy users, and adds the secret decryption information given by these users to the ciphertext. Thus, only t other partial decryption values, given by the target set of receivers, will be enough to correctly complete the decryption.

The primitive, however, was first formalized by Delerablee et al. [33]. They present an efficient construction that is fully dynamic and secure in the standard model. Besides, it is the first one achieving constant-size ciphertexts. Intuitively, the construction employs an arbitrary bilinear map group system. Let $e : G_1 \times G_2 \rightarrow G_T$ be a bilinear map, and g and h be two randomly selected generators for the groups G_1,G_2, respectively. In the scheme, the encryption key EK consists of $(2n + 2)$ group ele-

ments $(u, v, \{h^{\alpha.\gamma^i}\}_{i=0}^{2n-1})$ where $u \in G_1$, $h^{\alpha.\gamma^i} \in G_2$, and $v \in G_T$, and $(n-1)$ random integers $(d_1, ..., d_{n-1})$ from Z_p [1]. In addition, the combining key includes $(n-1)$ group elements $(h, h^{\gamma}, ..., h^{\gamma^{n-1}})$. Besides, the secret key of user j is $g^{\frac{1}{\gamma+j}}$.

The encryption algorithm computes the header and the session key for a threshold t of users as $(Hdr = (C_1 = u^{-k}, C_2 = h^{k\alpha P(\gamma)}), K = v^k)$ where $P(\gamma) = \prod_{x=1}^{n}(\gamma + x) \cdot \prod_{x \in D_1}(\gamma + x)$, $D_1 \subset D$, and $|D_1| = t - 1$. Note that for a threshold t, the algorithm adds $(t-1)$ dummy users to fix the degree of the polynomial $P(\gamma)$ as $(n + t - 1)$. Since the decryption procedure applies bilinear maps to C_2 and usk, a collection of t secret keys can be used collectively to reduce the degree of $P(\gamma)$ by t as they will have t different $(\gamma + x_i)$ in the denominator of the exponent [2]. After that, the combining key $\{h^{\gamma^i}\}_{i=0}^{n-2}$ will be enough to complete the decryption of the ciphertext. However, with fewer than t secret keys, the polynomial in the exponent of the $e(g, h)$ must have degree no smaller than n. As it is impossible to reconstruct $e(g, h)^{\gamma^n}$, the combining key will not be sufficient to complete the decryption of the given ciphertext. This works for any t, which is the core idea allowing a dynamic threshold.

An important feature of the scheme is that, all elements in the combining key are used in decrypting even a ciphertext with the threshold 1. For some applications this fact obstructs to utilize the primitive as an efficient building block. Besides, requiring the combining key seems to be unavoidable in construction of dynamic threshold public key encryption (DTPKE) with small size ciphertexts.

Our Results. We propose a new DTPKE scheme that has an allowed threshold

[1] The integers $(d_1, ..., d_{n-1})$ forms a set D of *dummy users*, which will be combined with the set of users in order to be consistent with the threshold values.

[2] The decryption procedure uses a special aggregate technique, which employs a simple fact that a product of inverses of coprime polynomials can be written as a sum of inverses of affine polynomials. For the detailed description, we refer the paper. [33]

range in the public parameters. Knowing the threshold range, our new construction uses a degree reduction technique on the exponent, and yields a combining key that scales linearly only in the size of the threshold range rather than the number of all users n [3]. This feature enables us to efficiently utilize the primitive in order to obtain an multi-secret sharing scheme, that we'll explain in the next chapter. Our scheme also achieves constant size ciphertexts. However, in contrast to [33], it does not allow the sender to dynamically choose the authorized set for each ciphertext; it fixes the target set at the setup.

3.1 Definition of DTPKE

The notion of dynamic threshold public key encryption (DTPKE) was proposed in [33] as a refinement of the regular threshold encryption scheme. In a DTPKE, each user has a secret key, and a sender can specify a threshold t so that the ciphertext requires t secret keys to jointly recover the plaintext. Moreover, the choice of threshold is not fixed, but can be arbitrary based on the sender's interests. Formally, a DTPKE is defined with the following algorithms:

- **D.Setup**(λ)**:** This algorithm takes a security parameter λ as input, and calculates a master secret key MK, an encryption key EK, and a combining key CK. It then outputs the public parameters $params = (EK, CK)$, and gives MK to the registration authority.

[3]Note that this change of having an allowed range is a strict generalization of the previous definition of DTPKE rather than a weakening. To see this, we can simply set the allowed range to be $[1, n]$ to instantiate the previous scheme [33].

- **D.KeyGen**(MK)**:** This algorithm takes MK as input, and outputs the user's secret key usk_i and the user's public key upk_i for each user i.

- **D.Enc**(EK, t, m)**:** This algorithm takes the encryption key EK, a threshold t, and a message M as inputs. It then outputs a ciphertext C.

- **D.ShareDecrypt**(usk_i, C)**:** It takes the user's secret key usk_i and a ciphertext C as inputs, and outputs a decryption share σ_i.

- **D.Combine**$(CK, T, \sigma_1, ..., \sigma_t, C)$**:**The algorithm takes the combining key CK, a ciphertext C, a subset T of t user pubic key, and a list $(\sigma_1, ..., \sigma_t)$ of t decryption shares as inputs. It either outputs a message M or $\perp$.

Note that Delerablee et al. [33] counted two more algorithms **ValidateCT** and **VerifyShare** as parts of usual DTPKE scheme. Since those algorithms can easily be obtained using traditional techniques, we omitted them here.

Correctness: The correctness of a dynamic threshold public-key encryption scheme requires that for any ciphertext $C = D.Enc(EK, t, M)$ with a threshold t, if t users correctly produced the partial decryption shares σ_i, then the **Combine** algorithm on the set $\{\sigma_1, ..., \sigma_t\}$ correctly outputs the message M. Formally,

$$\Pr[\textbf{D.Combine}(CK, T, \{\sigma_i\}_{i\in[t]}, C) = M] = 1$$

if $C = \textbf{D.Enc}(EK, t, M)$, and $\forall i \in [t], \sigma_i = \textbf{D.ShareDecrypt}(usk_i, C)$

Security: Briefly, the semantic security of the scheme guarantees that less than t users cannot decrypt a ciphertext with the threshold t. Even if we allow an adversary to corrupt $t - 1$ users, we claim that the content of the ciphertext with the threshold

t will still be secure. Consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- The adversary $\mathcal{A}$ submits a threshold t.
- The challenger simulates the **Setup** algorithm and gets the system parameters (MK, CK, EK). It then gives the public parameters $params = (EK, CK)$ to the adversary.
- $\mathcal{A}$ is allowed to make corrupt queries. To respond the queries, the challenger runs **D.KeyGen** to get secret key-public key pairs and sends the corresponding keys $(upk_1, usk_1), \ldots, (upk_q, usk_q)$ to $\mathcal{A}$.
- $\mathcal{A}$ submits two messages M_0 and M_1 with the threshold t. The challenger randomly choose $b \in \{0, 1\}$, and sends C_b as an encryption of M_b to $\mathcal{A}$.
- $\mathcal{A}$ can adaptively make extra q' corrupt queries.
- Finally, $\mathcal{A}$ outputs a guess b' and wins the game if $b' = b$.

We require in the above game that $q + q' < t$. The advantage of $\mathcal{A}$ is defined as $Adv^{cpa}_{\mathcal{A}} = |\Pr[b' = b] - \frac{1}{2}|$. We say a dynamic threshold public key encryption scheme is semantically secure if for any polynomial time adversary $\mathcal{A}$ the advantage $Adv^{cpa}_{\mathcal{A}}$ is a negligible function over λ.

Remark that the adversary can also make ShareDecrypt query, and get some decryption share of a ciphertext for some particular users. However, the corrupt queries give more advantages in comparison to ShareDecrypt queries, and even we we allow the adversary to corrupt up to t users in proving the security of the ciphertext with

the threshold t. Thus, allowing only corrupt queries would be enough for the security definition.

3.2 A New Construction of DTPKE

We here propose a new construction for DTPKE so that the combining (and encryption) key grows linearly in the *range* of the threshold t. The new scheme permits senders to specify an arbitrary threshold for the ciphertext in a given range $[\delta_0, \delta_1)$, and—as in [33]—, and achieves optimal size ciphertexts (a constant number of group elements). Additionally, our scheme is algebraically simpler than the original scheme of [33].

In more detail, the new scheme encodes the evaluation of a $(\delta_1 - 1)$-degree polynomial $P(x)$ evaluating at a secret point γ into a secret key $usk = g^{P(\gamma)}$. For a random k, α, the encryption algorithm encodes the evaluation of a $(t - \delta_0)$-degree polynomial $A(x)$ into a ciphertext $C_2 = h^{k\alpha A(\gamma)}$. During decryption, pairing a secret key with the ciphertext C_2 would form a decryption share $e(g, h)^{k\alpha A(\gamma)P(\gamma)}$ which will have the evaluation of a $(\delta_1 - \delta_0 + t - 1)$-degree polynomial at γ in the exponent. The essential point for correctness here is to remove the dependence between the decryption share and the threshold t in their exponent. We observe that using simple linear algebra, t users can find t coefficients to remove $(t - 1)$ higher terms of $P(x)A(x)$, arriving at a polynomial with degree $(\delta_1 - \delta_0)$; with one more small trick, we can bring it down to $\delta_1 - \delta_0 - 1$. Thus, the combining key containing $(\delta_1 - \delta_0 - 1)$ corresponding powers will be enough to complete the decryption. While for the security, if there are not enough decryption shares, there is always some leftover term in the exponent to mask

the session key. Formally, given a threshold interval $[\delta_0, \delta_1)$, our DTPKE is as follows:

- Setup: The algorithm first chooses parameters defining a bilinear map system for a given security parameter λ: $\mathcal{B} = (p, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e)$ where $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ is a bilinear map and $|p| = \lambda$. The algorithm then randomly chooses two integers $\gamma, \alpha \in \mathcal{Z}_p^*$, and two generators $g \in \mathcal{G}_1$ and $h \in \mathcal{G}_2$. It outputs the master secret key $msk = (g, \alpha, \gamma)$, the encryption key EK $= (u, v, \{h^{\alpha.\gamma^i}\}_{i=0}^{\delta_1-\delta_0-1})$, and the combining key CK $= \{h^{\gamma^i}\}_{i=0}^{\delta_1-\delta_0-1}$ where $u = g^{\alpha.\gamma}$ and $v = e(g, h)^{\alpha}$.

- **KeyGen**:(i, msk) The algorithm takes the master secret key msk as input. It assigns an integer $i \in Z_p^*$ to each user as his upk, and constructs the $(\delta_1 - 1)$-degree polynomial $P_i(x) = i^{\delta_1-1}x^{\delta_1-1} + ... + i^2x^2 + ix + 1$. It then outputs the user's key pairs as $usk = g^{P_i(\gamma)}$ and $upk = i$ for all users.

- **Enc**$(t, \delta_0, \delta_1, EK, M)$: The algorithm takes as input the encryption key EK, a threshold value $t \in [\delta_0, \delta_1)$, and a message M. It randomly chooses an integer $k \in Z_p^*$ and calculates the ciphertext $C = (C_1, C_2, C_3)$ as

$$C_1 = u^{-k}, \qquad C_2 = h^{k\alpha A(\gamma)}, \qquad C_3 = K \cdot M,$$

  where the polynomial $A(x) = (x + 1) \dots (x + t - \delta_0)$ and $K = v^k$ is used as a session key. Since $A(\cdot)$ is $(t - \delta_0)$-degree polynomial, and $t \in [\delta_0, \delta_1)$, the second component $C_2 = h^{k\alpha A(\gamma)}$ of the ciphertext can be computed from EK (see the explanation below).

- **ShareDecrypt**(usk_i, C): The algorithm takes a ciphertext (C_1, C_2, C_3) and

secret key of a user as inputs. It computes the decryption share as

$$\sigma_i = e(usk_i, C_2) = e(g,h)^{k.\alpha \cdot P_i(\gamma) \cdot A(\gamma)}.$$

- **Combine**$(CK, \{\sigma_i\}), \{upk_i\})$: This algorithm takes the set of decryption shares $\{\sigma_1, \ldots, \sigma_t\}$ and the identities $\{1, \ldots, t\}$ as inputs (w.l.o.g, we assume that the shares are from users $1, \ldots, t$, and the $P_i(\cdot)$ are defined using the coefficients $\langle 1, i, i^2 \ldots, i^{\delta_1 - 1} \rangle$ for $i \in [t]$).

  It first finds t values $a_1, \ldots, a_t$, (not all zero) that satisfy $\sum_{i=1}^{t} a_i P_i(x) = Q(x)$, where $Q(x)$ has the $(t-1)$ higher degree term to be 0 (thus $Q(x)$ is with degree $(\delta_1 - t)$). This can be done by solving the following linear system:

$$\begin{bmatrix} 1 & 2^{\delta_1 - 1} & \cdots & t^{\delta_1 - 1} \\ 1 & 2^{\delta_1 - 2} & \cdots & t^{\delta_1 - 2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{\delta_1 - t + 1} & \cdots & t^{\delta_1 - t + 1} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.2.1}$$

  It follows that: $\prod_{i=1}^{t} (\sigma_i^{a_i}) = e(g,h)^{k.\alpha \cdot A(\gamma) \cdot \sum_{i=1}^{t} a_i Pi(\gamma)}$, and it is equal to $e(g,h)^{k\alpha \cdot Q(\gamma) A(\gamma)}$ for a $(\delta_1 - \delta_0)$-degree polynomial $Q(x)A(x)$.

  We further define:

$$R(x) = \frac{Q(x) \cdot A(x) - H}{x}, \text{ where } H = \sum a_i \cdot (t - \delta_0)!$$

In one formula, the combine algorithm then extracts the key as:

$$\begin{aligned} K &= [e(C_1, h^{R(\gamma)}) \prod_{i=1}^{t} \sigma_i^{a_i}]^{1/H} \\ &= [e(g^{-k\alpha\gamma}, h^{R(\gamma)}) e(g,h)^{k\alpha Q(\gamma)\cdot A(\gamma)}]^{1/H} \\ &= e(g,h)^{k\alpha[Q(\gamma)A(\gamma) - \gamma R(\gamma)]/H} \\ &= e(g,h)^{k\alpha} \end{aligned}$$

(All the above value can be computed from public values and see more details below about correctness).

Finally, it outputs C_3/K as the plaintext.

Correctness: Note that all the coefficients of the polynomial $A(x)$ are public and $A(x)$ is with degree $t - \delta_0 < \delta_1 - \delta_0$, thus the ciphertext $C_2 = (h^{\alpha A(\lambda)})^k$ is computable from the public values $h^\alpha, h^{\alpha\gamma}, \ldots, h^{\alpha\gamma^{\delta_1 - \delta_0 - 1}}$ contained in EK.

Furthermore, since the degree $(\delta_1 - 1)$ polynomials $\{P_i(x)\}$ are also public, the equations for the linear system can be formed. Note that the matrix has rank $(t-1)$ and there are t unknowns for the equations, from linear algebra, we can always find a solution set for $a_1, \ldots, a_t$. More importantly, the linear system aims at reducing the degree of $\sum_i a_i P_i(x)$, i.e., to kill the higher degree terms, thus the resulting polynomial $Q(x)$ will have degree $[\delta_1 - 1 - (t-1)] = \delta_1 - t$.

Finally, the polynomial $R(x) = \frac{Q(x)\cdot A(x) - H}{x}$, where $H = \sum a_i \cdot (t - \delta_0)!$ is the constant term of $Q(x)A(x)$, thus $R(x)$ will have degree $\delta_1 - \delta_0 - 1$, and all the coefficients are known (or defined by the values of $\langle a_1, \ldots, a_t \rangle$). It follows that $h^{R(\gamma)}$ can be computed from the public values $h, h^\gamma, \ldots, h^{\gamma^{\delta_1 - \delta_0 - 1}}$ that are contained in the combining key.

Efficiency: We can see that the construction has secret key and ciphertext both given by a constant number of group elements; the encryption key and combining key size grow linearly in the size of the range where the thresholds fall.

Security: We first briefly explain the intuition for the security. In the **KeyGen** algorithm, the rows of Vandermonde matrix are used as the coefficients for generating the polynomial $P_i(x)$ in the secret keys. Since any square sub-matrix of the Vandermonde matrix has full rank, the system (1) with t equations will have only one solution (in which all a_i are zero). However, the system (1) with at most $(t-1)$ equations and t coefficients a_i will have many solutions—one will be nonzero. Thus, when $t < \delta_1$, t users can decrease the degree on the exponent of the decryption shares by at most $t-1$. When the ciphertext is assigned threshold $t' > t$, the leftover degree (on the exponent hiding K) will be larger than $\delta_1 - \delta_0$, which can not be computed from CK (otherwise, it means the adversary can reduce degrees on exponent and violates the underlying assumption).

Theorem 1. The dynamic threshold public key encryption scheme given above is IND-CPA secure under MSE-DDH assumption.

Proof. Assume there is an adversary $\mathcal{A}$ that breaks the security of DTPKE, then we claim that we build a simulator $\mathcal{S}$ that breaks the security of the assumption MSE-DDH. The algorithm $\mathcal{S}$ interacts with the adversary $\mathcal{A}$ as follows:

- $\mathcal{A}$ first submits a threshold t^*. The simulator then gives the parameters $\ell = (\delta_1 - \delta_0)$, $n = \delta_1$, and $t = (t^* - \delta_0)$ to the challenger of the assumption MSE-DDH, and gets an MSE-DDH instance. So, the simulator has two coprime polynomials f and g with the degree $(\delta_1 - \delta_0)$ and $(t^* - \delta_0)$ respectively, and

with their distinct roots $y_1, \ldots, y_{(\delta_1-\delta_0)}$ and $z_1, \ldots, z_{(t^*-\delta_0)}$. $\mathcal{S}$ has also a sequence of exponentiations

$$g_0, g_0^{\gamma}, \ldots, g_0^{\gamma^{2.\delta_1-\delta_0-1}}, \qquad g_0^{k.\gamma.f(\gamma)},$$
$$g_0^{\alpha}, g_0^{\alpha.\gamma}, \ldots, g_0^{\alpha.\gamma^{2.\delta_1-\delta_0-1}},$$
$$h_0, h_0^{\gamma}, \ldots, h_0^{\gamma^{\delta_1-1}},$$
$$h_0^{\alpha}, h_0^{\alpha.\gamma}, \ldots, h_0^{\alpha.\gamma^{\delta_1-1}}, \qquad h_0^{k.g(\gamma)}$$

as well as $T \in \mathcal{G}_T$ which is either equal to $e(g_0, h_0)^{k.f(\gamma)}$ or some random element of $\mathcal{G}_T$.

- The simulator $\mathcal{S}$ first chooses $(\delta_1 - t^* - 1)$ random integers x'_j where $j \in [\delta_1 - t^* - 1]$, and sets the public values $x_i = z_i$ for $i \leq (t^* - \delta_0)$ and $x_{t^*-\delta_0+i} = x'_i$. It then sets $g = g_0^{f(\gamma)}$ and $h = h_0$, and gives the encryption key $\mathrm{EK} = (u, v, \{h^{\alpha.\gamma^i}\}_{i=0}^{\delta_1-\delta_0-1}, \{x_j\}_{j=0}^{\delta_1-\delta_0-1})$ and the combining key $\mathrm{CK} = (\{h^{\gamma^i}\}_{i=0}^{\delta_1-\delta_0-1})$ where $u = g^{\alpha.\gamma}$ and $v = e(g, h)^{\alpha}$ to the adversary. Note that the simulator can compute u and v from the given instance.

- When the adversary makes a corrupt query for a user with the identity id, the simulator first picks a random integer $i \in [n]$, and gives the corresponding secret key $usk_i = g^{P_i(\gamma)}$ and public key $upk_i = i$ to the adversary. Note that since $P_i(x) = i^{\delta_1-1}.x^{\delta_1-1} + ... + i^2.x^2 + i.x + 1$ is an $(\delta_1 - 1)$-degree polynomial, the secret key $usk_i = g_0^{f(\gamma).P_i(\gamma)}$ can be computed from the given instance.

- The adversary submits two messages M_0, M_1. The simulator chooses a bit

$b \in \{0, 1\}$, and sets the ciphertext $C_b = (C_1, C_2, C_3)$ as

$$C_b = (g_0^{-k.\gamma.f(\gamma)}, h_0^{k.g(\gamma)}, T \cdot M_b)$$

and returns C_b to $\mathcal{A}$. It can be easily proved that if we set $k' = k/\alpha$, then

$$C_1 = u^{-k'}, \quad C_2 = h^{k'.\alpha.g(\gamma)}.$$

- The adversary outputs a guess b' and the simulator gives b' to the challenger.

Assume $Adv_{\mathcal{A}}^{DTPKE}$ is the advantage of $\mathcal{A}$ in breaking the security of the construction, and $Adv_{\mathcal{S}}^{MSE-DDH}$ is the advantage of $\mathcal{S}$ in breaking the security of the assumption. If $T = e(g_0, h_0)^{k.f(\gamma)}$, then $\mathcal{A}$' s view will be identical to a real attack. Thus $Adv_{\mathcal{A}}^{DTPKE} = |\Pr[b' = b] - 1/2| > \epsilon$. However, if T is just random group element, then $\Pr[b' = b] = 1/2$. Therefore

$$Adv_{\mathcal{S}}^{MSE-DDH} \geq | \frac{1}{2} \pm \epsilon - \frac{1}{2} | = \epsilon.$$

□

Chapter 4

Graded Secret Sharing

Many cryptographic systems rely on a trusted authority that maintains and employs a secret to carry out basic tasks such as encryption, decryption, or key generation. Reliance on a single authority, however, raises immediate security and reliability concerns. Secret-sharing schemes can be used to overcome these obstacles. Given a set of n parties and a collection $\mathcal{A}$ of subsets of the parties, a *secret sharing scheme* for $\mathcal{A}$ is a method of distributing shares of a secret among the parties so that any subset of parties from $\mathcal{A}$ can reconstruct the secret, but any subset of parties which is not in $\mathcal{A}$ cannot. It was introduced by Blakley [10] and Shamir [80] for threshold structures: in such "threshold schemes," any subset of the parties can recover the secret precisely when the cardinality of that particular subset is above a certain threshold. Ito et al. [71] later generalized the notion to more general access structures $\mathcal{A}$.

A fundamental efficiency concern for secret sharing schemes is the size of the share that must be distributed to each participant. Karnin [64] showed that in any unconditionally secure secret sharing scheme, the length of each share must be at least

that of the secret. To achieve better efficiency, Krawczyk [66] relaxed the security guarantee by introducing a natural computational notion: while t shares can be used to efficiently recover the secret, any fewer than t give no information about the secret to a computationally bounded adversary. Each share in the scheme has size $(m/t+\lambda)$ where λ is the security parameter and m the size of the secret to be shared.

An important and well-studied variant arises when multiple secrets must be distributed to the parties [5,13,62,87,89]; such schemes are known as *multi-secret sharing schemes (MSSS)*. In a $(t_1, \ldots, t_\ell)$-*multi-secret sharing scheme* (MSSS), ℓ independent secrets $s_1, \ldots, s_\ell$ are shared with n parties in such a way that at least t_i parties are required to recover the secret s_i. For simplicity, we will assume $\ell \leq n$ throughout the paper, and sometimes we use them interchangeably.

Blundo et al. [13] established that in order to achieve unconditional security for multi-secret threshold schemes, the size of the share that each participant holds must be, at least, linear in the number of secrets. Besides, Masucci [73] presented a weaker notion of security for multi-secret sharing schemes in unconditional settings, and gave some lower bounds for the size of the shares of each participant. Herranz et al. [56] proved that the size of the share for each participant in multi-secret threshold schemes must be linear in the number of secrets, even if the scheme enjoys the weaker unconditional security introduced by [73].

Motivated by Krawczyk's success in the single secret case, computationally secure MSSSs are considered for better efficiency [55, 56]. However, straightforward application of Krawczyk's scheme [66] for a MSSS still incurs a linear size share for each player, e.g., in [55], ℓ instances of [66] are run and the resulting share size is $m(\frac{1}{t_1} + \ldots + \frac{1}{t_\ell}) + \ell\lambda$ (if there are ℓ secrets). The first part is inherent due to the message, while the second part as overhead grows linear in number of secrets. Her-

ranz et al. [56] further observe that in the computational setting, one can shift the private storage to the public via encryption, i.e., each player encrypts his shares using a "symmetric key encryption", and he only has to store a short key. In particular, an information theoretic MSSS can be used to generate the shares first, then the shares are encrypted and stored to a "public bulletin board".

Introducing such public bulletin board indeed dramatically reduces the private storage, however, it is not preferable for several reasons: (i) some trusted third party needs to be always online to provide such public bulletin board service[1] while in many applications of secret sharing scheme, the system is decentralized and no such service might be available, (for instance, our application to secure multiparty computation in section 4.4) (ii) the public storage for each player is as large as in the information theoretic case, ($\ell \cdot m$ for each player at least [56]) thus despite the small amount of private storage needed it is still open to achieve small storage per player overall (both public and private)[2].

Our Results.

We present a new construction for threshold MSSS (we here focus entirely on threshold MSSSs, so we drop the adjective "threshold") that achieves share size $O(m(\frac{1}{t_1} + \ldots + \frac{1}{t_\ell}) + \log \ell \cdot \lambda)$ which is close to optimal when ℓ is large and does not require any "public data". As the first part is inherent for carrying the information of the message, the second part is the overhead that we try to minimize. A very

[1]The need for such storage should be distinguished from the need for public parameters, e.g., the description of an algebraic group. The public parameter can be set up once and distributed to every party without augmenting the share size (since such parameters are very short).

[2]One may wonder whether simply applying the tool of information dispersal (IDS) on the data stored in the public bulletin board solves the problem, however, doing so is highly non-trivial. As we will show later, in order to make the IDS effective, we first need to carefully design the scheme so that the public bulletin board size scales only linear with each threshold, not always ℓ as in [56].

natural parameter choice which we will use for the application to secure multiparty computation (MPC), we consider a $(1, 2, \ldots, \ell)-$MSS. In such a scheme, the total share size for each user is $O(m(1 + \frac{1}{2} + \ldots + \frac{1}{\ell}) + \log \ell \cdot \lambda) = O(\log \ell \cdot (m + \lambda))$.

We proceed in two steps: we first give a natural, generic construction of a (threshold) multi-secret sharing scheme from a dynamic threshold public key encryption scheme (DTPKE). (The detailed definition of this primitive is described in the chapter 3); we then give a recursive adaptation of the construction using the new construction of DTPKE (that we explained in the previous chapter), that yields a new MSSS with nearly optimal share size in the case that the number of secrets to be distributed is large.

The generic construction. As in [66], we use an encryption scheme as a building block. To capture the various threshold requirements, we rely on a *dynamic threshold public key encryption scheme (DTPKE)* [33] which enables a sender to dynamically set a threshold value for the ciphertext for the decryption capability among the users. Intuitively, given the secrets $(s_1, .., s_\ell)$, we apply a DTPKE scheme to encrypt each secret s_i with its threshold t_i; thus, the resulting ciphertext c_i requires at least t_i users in order to be decrypted. Then, following [66], we apply an efficient *information dispersal scheme (IDS)* [66, 78] to distribute the ciphertexts together with all the public keys among users in a way that t_i users can construct c_i from their shares. This yields a generic construction that transforms a DTPKE scheme into an MSSS. The resulting construction is computationally secure if the given DTPKE is semantically secure.

Recursive application of the generic construction. As we emphasized above, we here aim to get an efficient MSSS that does not require any public data. So, we

need to distribute all public parameters among the participants, even the combining key that the DTPKE schemes possess. On the other hand, since all components of the combining key are required in decrypting even a ciphertext with threshold 1, all combining key should be given to each participant. It is clear that, even with our new construction of DTPKE, applying our general reduction straightforwardly results in a linear size share for each participant. To circumvent this obstacle, we divide the threshold range into $\log n$ parts: $\Delta_1, \ldots, \Delta_{\log n}$, where $\Delta_u = [2^{u-1}, 2^u)$, (W.l.o.g, we assume n is a power of 2). We then run an independent DTPKE instances together with an IDS for each part Δ_u in order to share the secrets $\{s_i | t_i \in \Delta_u\}$. For each window Δ_u, there are at most 2^u elements for the combining key, while the minimum threshold for this range is 2^{u-1}, thus we can now apply the information dispersal scheme [66, 78] to efficiently distribute the combining key, i.e., each share with size $O(1)$. Since $n = \text{poly}(\ell)$, the final share size is $O(\log \ell)$, where ℓ is the number of secrets.

4.1 Definition

In this section, we give the definition and the security requirements of the multi-secret threshold schemes. We here formalize $(t_1, \ldots, t_\ell)$-multi-secret threshold sharing scheme such that each t_i is the threshold value for the corresponding secret s_i, and $t_i \in [n]$ where n is the number of users. However, we consider $(1, \ldots, n)$-MSSS, which we call graded secret sharing, for the application of secure multiparty computation (MPC) that we discuss at the end of this chapter.

Definition 1. A $(t_1, \ldots, t_\ell)$-multi-secret threshold sharing scheme is defined by two

protocols as follows:

- **Share:** This protocol takes the security parameter λ, the set of n players $\mathcal{P}$, the global secret $\overrightarrow{s} = (s_1, ..., s_\ell)$ to be shared, and ℓ threshold values $(t_1, ..., t_\ell)$ where $t_i \in [n]$. It outputs the set of shares $\{sh_i\}_{P_i \in \mathcal{P}}$.

- **Reconstruct:** This is a protocol executed among a subset of users: each user i takes the index $j \in [\ell]$ and his secret share sh_j as input, and they jointly output the corresponding secret s_j or $\perp$.

Remark that the share protocol also outputs some public parameters required for the reconstruct protocol (such as description of the underlying group system). Since we consider them as part of share for each user, we didn't specify them as separate output.

A multi-secret sharing scheme has to satisfy the following two properties:

Correctness: Enough shares are able to reconstruct the secret. Formally, if $S = \{sh_{i_k}\}_{k=1}^{t_j} \subseteq \{sh_i\}_{P_i \in \mathcal{P}}$ where $\text{Share}(\lambda, \mathcal{P}, \overrightarrow{s}, \{t_i\}) = (\{sh_i\}_{P_i \in \mathcal{P}})$, then

$$s_j \leftarrow \text{Reconstruct}(j, \{sh_{i_k}\}_{k=1}^{t_j}).$$

However, if $|S| < t_j$, then $\text{Reconstruct}(j, S) = \perp$.

Security: We define the computational security of a multi-secret threshold sharing scheme similar to the regular security notion of semantic security that is used in encryption schemes. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- The challenger gives the set $\mathcal{P}$ of users and the thresholds values $(t_1, ..., t_\ell)$ to

the adversary.

- The adversary $\mathcal{A}$ submits the challenge threshold t_i.

- The adversary also submits two global secrets $\overrightarrow{s}^{(0)}, \overrightarrow{s}^{(1)}$ where $\overrightarrow{s}^{(0)} \neq \overrightarrow{s}^{(1)}$ and $s_j^{(0)} = s_j^{(1)}$, $\forall j \neq i$.

  For simplicity, we require the adversary to submit two global secrets $\overrightarrow{s}^{(0)}, \overrightarrow{s}^{(1)}$ such that they differ only at the index i. It can be easily seen to imply the general case.

- The challenger chooses a random bit $b \in \{0, 1\}$, and runs the share protocol $(\{sh_k\}_{P_k \in \mathcal{P}}, params) \leftarrow \text{Share}(\overrightarrow{s}^{(b)}, \mathcal{P}, \{t_k\}, \lambda)$, and gives $(\{sh_k\}_{k<t_i}, params)$ to the adversary.

- Finally, $\mathcal{A}$ outputs a guess b'.

The advantage of the adversary in breaking the security of the scheme is determined as

$$Adv_{\mathcal{A}}^{MSSS} = |Pr[b' = b] - 1/2|.$$

We say that the scheme is computationally secure if $Adv_{\mathcal{A}}^{MSSS}$ is negligible.

4.2 Generic Construction of Computational Multi-secret Sharing Schemes

To circumvent the lower bound restriction in the perfect secret sharing scheme [64], Krawczyk [66] proposed a computational (t, n)-secret sharing for one secret. Specif-

ically, after encrypting the secret, the ciphertext is distributed via an information dispersal scheme to the users, while a regular Shamir secret sharing scheme is applied to distribute the secret key. It can achieve essentially optimal share size: m/t, where m is the size of the data to be shared and t is the threshold. As a secret key is only λ bits long, the final share size for each user is only $m/t + \lambda$. This beautiful framework inspires us to consider the efficiency benefits of a computational notion of security in the setting of multi-secret sharing as well.

A natural way to generalize Krawczyk's idea to multi-secret sharing (assuming there are ℓ independent secrets to share) is to run the Krawczyk scheme ℓ times. Each message can be encrypted under a different key, and the ciphertexts will be dispersed as in [66]; ℓ independent Shamir secret sharing schemes are then carried out to distribute the keys. Thus the total size will be $\sum_{i=1}^{\ell}(m_i/t_i + \lambda) = \sum_{i=1}^{\ell} m_i/t_i + \ell\lambda$. This strategy was observed in [55] and analyzed there in detail. As discussed above, our goal is to construct a multi-secret sharing scheme with sub-linear overhead in the number of secrets, that does not require a "bulletin board".

It seems that the above strategy inherently requires overhead linear in ℓ, as each message and secret key are treated independently. We need to deviate from this natural construction and generalize Krawczyk's idea in a more elaborate way. Observe that if each user had only one single secret key, this might alleviate the overhead. To realize this plan, we apply a special type of (threshold) encryption scheme so that the ciphertext can be created with flexible thresholds while each user has only a single secret key.

4.2.1 Constructing MSSS from DTPKE

We now show how to use the DTPKE to construct a MSSS that provides computational security. This generic construction will be a stepping stone for our final efficient construction and will permit us to treat the security analysis in a modular fashion.

First, consider an intermediate case where there is a bulletin board through which all data can be accessed by any user. Assume there are ℓ secrets $s_1, ..., s_\ell$ to be shared with corresponding thresholds $t_1, ..., t_\ell$. These secrets can simply be encrypted using the DTPKE with the corresponding thresholds, and the resulting ciphertexts—along with the combining keys—can be placed on the bulletin board. In addition, each user is given a secret key as a share. To reconstruct the secret s_i, t_i users jointly decrypt the corresponding ciphertext and get the plaintext as the secret s_i. However, we aim to obtain an efficient scheme that does not include a bulletin board. So, we need to distribute the bulletin board among users. The ciphertexts can be distributed via an IDS without affecting the security. But all the combining keys are required to be able to decrypt even a ciphertext with the threshold 1. Thus, all of them should be given to each user.

Formally, given an information dispersal algorithm IDS, and a DTPKE scheme **D=(D.Setup, D.KeyGen, D.Enc**, **D.ShareDecrypt, D.Combine**), a multi-secret threshold sharing scheme can be obtained as follows:

- **Share**: The algorithm takes the security parameter λ, the set of users $\mathcal{P} = \{1, \ldots, n\}$, the secrets $s_1, ..., s_l$ to be distributed, and the threshold values $\{t_i\}_{i\in[\ell]}$. It first calls **D.Setup**(λ) to get the parameters (MK, EK, CK).

  It then runs **D.KeyGen** algorithm to get the secret keys and the public keys $\{upk_j, usk_j\}_{j=1,...,n}$ for the users.

The algorithm then encrypts each secret and computes $c_i = \mathbf{D.Enc}(EK, t_i, s_i)$ for the secrets s_i using its threshold t_i.

Next, the algorithm distributes each c_i by running an IDS on it using threshold t_i. Suppose the shares are $\{c_{i,j}\}_{i\in[\ell],j\in[n]}$.

This algorithm ends with each user j receiving his key pair (usk_j, upk_j), the combining key CK, and ℓ shares $(c_{1,j}, ..., c_{\ell,j})$ for ciphertexts.

- **Reconstruct**: Assume there is a set S of users that jointly run the protocol in order to get the corresponding secret s_k, and each user holds a share $(usk_j, upk_j, CK, \{c_{i,j}\})$ where $j \in S$, $i \in [\ell]$, and $|S| \geq t_k$. They first jointly reconstruct c_k using the IDS reconstruct algorithm on the shares $\{c_{k,j}\}_{j\in S}$. Then each user in the set S runs **D.ShareDecrypt** on c_k to get a decryption share $\sigma_{k,j}$. Finally, they jointly compute the secret $s_k = \mathbf{D.Combine}(\{\sigma_{k,j}\}, c_k, CK)$.

Remark. Regarding efficiency, each user's share has size $|CK| + |usk| + |upk| + \sum_{i=1}^{l} |c_{i,j}|$. In order to guarantee sub-linear size share, we need to find an efficient IDS that results in sublinear size shares of ciphertexts. Besides, we demand a dynamic threshold encryption scheme that, either the combining key of the scheme is constant, or the structure of the scheme enables us to employ IDS on CK such that each share size of CK will be at most sub-linear.

Correctness. First according to the IDS property, when $|S| \geq t_k$, c_k can be recovered; then following the correctness of the underlying DTPKE, $|S|$ secret keys together with the combing key can jointly decrypt s_k.

Theorem 2. If **D** is a semantically secure DTPKE scheme, then the construction given is a computationally secure MSSS.

Proof. Assume there is an adversary that breaks the security of the multi-secret threshold scheme; then we can build an algorithm $\mathcal{B}$ that breaks the security of the scheme $\mathcal{D}$. Consider the following game between the algorithm $\mathcal{B}$ and the adversary $\mathcal{A}$.

- $\mathcal{B}$ submits the set of users $\mathcal{P}$ and ℓ threshold values $(t_1, \ldots, t_\ell)$.
- $\mathcal{A}$ submits the threshold t_{i^*} and two global secrets $\overrightarrow{s}^{(0)}$, $\overrightarrow{s}^{(1)}$ where $\overrightarrow{s}^{(0)}$ and $\overrightarrow{s}^{(1)}$ differ only at the index i^*.
- The algorithm $\mathcal{B}$ invokes the challenger of the scheme $\mathcal{D}$ in order to get the public parameters (EK, CK).

  $\mathcal{B}$ then makes corrupt queries for any q users, and gets q secret key-public key pairs $\{usk_j, upk_j\}_{j=1}^{q}$ from the challenger.

  $\mathcal{B}$ gives two secrets $s_{i^*}^{(0)}$ and $s_{i^*}^{(1)}$ together with the threshold value t_{i^*} to the challenger, and receives the challenge ciphertext $c_{i^*}^{(b)}$ of the secret $s_{i^*}^{(b)}$.

  $\mathcal{B}$ makes extra corrupt queries for any q' more users and gets q' secret key-public key pairs $\{usk_j, upk_j\}_{j=q+1}^{q+q'}$ where $q + q' = t_{i^*} - 1$.

  $\mathcal{B}$ also computes the ciphertexts c_i of the secrets $s_i^{(0)}$ for all $i \neq i^* \in [\ell]$. It then produces the shares $\{c_{i,j}\}_{i\in[\ell], P_j\in\mathcal{P}}$ for the ciphertexts $(c_1, \ldots, c_{i^*-1}, c_{i^*}^{(b)}, c_{i^*+1}, \ldots, c_l)$ by running IDS protocol.

  Finally, $\mathcal{B}$ gives all $(t_{i^*} - 1)$ shares $\{(usk_{j_k}, upk_{j_k}, CK, c_{i,j_k})\}_{i\in[\ell], k<t_{i^*}}$ to the adversary.

- The adversary $\mathcal{A}$ submits a guess b'. The algorithm $\mathcal{B}$ gives this guess b' to the challenger of DTPKE as a guess for the challenge ciphertext $c_{i^*}^{(b)}$.

Assume $Adv_{\mathcal{A}}^{MSSS}$ is the advantage of $\mathcal{A}$ in breaking the security of the construction, and $Adv_{\mathcal{S}}^{DTPKE}$ is the advantage of $\mathcal{S}$ in breaking the security of DTPKE. Since $\mathcal{A}$' s view will be identical to a real attack, if $Adv_{\mathcal{A}}^{MSSS} > \epsilon$, then $Adv_{\mathcal{S}}^{DTPKE} \geq \epsilon$. Thus, we don't have any security loss, and get very tight reduction.

□

4.3 Computational MSSS with Near Optimal Share Size

With the new construction of DTPKE developed in the chapter 3, we can construct an efficient multi-secret sharing scheme. Assume there are n users and ℓ secrets, and $n = \text{poly}(\ell)$. Intuitively, we group the secrets into $\log n$ classes according to their threshold ranges, i.e., $[1, 2)$,$[2, 4), \ldots, [n/2, n)$. For the secrets in the range of $[\delta/2, \delta)$, we will run an independent instance of our general construction of multi-secret sharing, instantiating the underlying DTPKE with the new scheme we introduce in section 3.2.

Note that the generic construction will be slightly altered: for each threshold range $[\delta/2, \delta)$, the combining key CK is dispersed using an IDS with threshold $\delta/2$. Since our new DTPKE for threshold range $[\delta/2, \delta)$ has the combining key with the size $O(\delta/2)$ (growing linearly with the size of the range instead of the largest threshold value n), running an IDS on this CK will yield share size $O(\delta/2)/(\delta/2) = O(1)$. Thus, we will have final share size $|C|(1/t_1 + 1/t_2 + \ldots + 1/t_\ell) + O(\log n)$ where $|C|$ is the ciphertext size of each message and $t_1, \ldots, t_\ell$ are the thresholds for each secret, respectively. The first term is essentially the smallest size corresponding to the amount of information in the secrets, and the second part is the overhead we are

aiming to minimize in this paper.

Given an efficient IDS (**IDA.Disperse, IDA.Rec**), and our new DTPKE, $\mathbf{PD} = \langle \mathbf{PD.Setup}, \mathbf{PD.KeyGen}, \mathbf{PD.Enc}, \mathbf{PD.ShareDecrypt}, \mathbf{PD.Combine} \rangle$, the details of our final construction of multi-secret sharing are presented as follows:

- **Share**: The algorithm takes the security parameter λ, the set $\mathcal{P}$ of n users, and the secrets $s_1, \ldots, s_\ell$ to be distributed as inputs. It first groups these secrets into $\log n$ groups $\Delta_1, \ldots, \Delta_{\log n}$, according to the threshold values, where $\Delta_u := [2^{u-1}, 2^u)$ for $u = 1, \ldots, \log n$. Here s_i with threshold t_i belongs to Δ_u if $t_i \in \Delta_u$. For each group Δ_u, the algorithm runs $\mathbf{PD.Setup}(\lambda)$ to generate parameters $\{(MK_u, EK_u, CK_u)\}_{u=1,\ldots,\log n}$.

  The algorithm then encrypts all the secrets using the corresponding master public keys and thresholds to generate the ciphertext $\{c_1, \ldots, c_\ell\}$, where $c_i = \mathbf{PD.Enc}(EK_u, t_i, s_i)$ if s_i is in group Δ_u.

  Now the algorithm runs the IDS on each ciphertext c_i using thresholds t_i, i.e., the shares of c_i are generated as $\{c_{i,j}\}_{j=1,\ldots,n} \leftarrow \mathbf{IDA.Disperse}(c_i, t_i)$.

  It also runs the IDS on each combining key CK_u using threshold 2^{u-1} and gets the shares $\{CK_{u,j}\}_{j=1,\ldots,n}$.

  For each user j, the algorithm runs **PD.KeyGen** on each range of thresholds to generate key pairs $(upk_{j,u}, usk_{j,u}) \leftarrow \mathbf{PD.KeyGen}(MK_u, j)$ for $j = 1, \ldots, n$. Also, it distributes the corresponding shares of ciphertexts and combining keys to this user.

  This algorithm ends with each user receiving his shares of ciphertexts $\{c_{i,j}\}_{i=1,\ldots,\ell}$, his shares of CK $\{CK_{u,j}\}_{u=1,\ldots,\log n}$, and his key pairs $\{(upk_{j,u}, usk_{j,u})\}_{u=1,\ldots,\log n}$.

- **Reconstruct**: A set S of t_k users jointly run this protocol. Each user $j \in S$ takes the index k and his share $\{(usk_{j,u}, upk_{j,u}), CK_{u,j}, c_{i,j}\}_{u\in[\log n]}$ as inputs. The users first jointly run the **IDA.Rec** protocol to reconstruct the ciphertext c_k and the combining key CK_u (assuming $t_k \in [2^{u-1}, 2^u)$) that were dispersed using thresholds $t_k, 2^{u-1}$ respectively, i.e., they jointly reconstruct $c_k \leftarrow \textbf{IDA.Rec}(\{c_{k,j}\}_{j\in S})$; and $CK_u \leftarrow \textbf{IDA.Rec}(\{CK_{u,j}\}_{j\in S})$.

  Next, each user j in the set S partially decrypts the ciphertext, i.e., $\sigma_{k,j} \leftarrow \textbf{PD.ShareDecrypt}(usk_{j,u}, c_k)$ to get decryption share $\sigma_{k,j}$.

  Finally, they jointly compute the secret $s_k \leftarrow \textbf{PD.Combine}(\{\sigma_{k,j}\}_{j\in S}, CK_u, c_k)$

  Each party outputs the secret s_k.

Efficiency: As we discussed at the beginning of the section, for each secret s_i with threshold t_i, suppose $t_i \in [\delta/2, \delta)$ where $\delta = 2^{u-1}$ for some $u \in [\log n]$. The ciphertext c_i corresponding to this secret will be dispersed using threshold t_i which will have size $|C|/t_i$ (using, e.g., Rabin IDS [78], where $|C|$ is the ciphertext size), thus the ciphertext shares together have size $|C|/t_1 + |C|/t_2 + \ldots + |C|/t_\ell$ assuming all the ciphertexts are the same length. Regarding the combining key shares, there are in total $\log n$ combining keys, each with size $|CK_u| = O(2^u)$ which will be dispersed using threshold 2^{u-1}; thus each contributes $O(1)$ to the share size and the total size from combining keys is $O(\log n)$. Additionally, there are $\log n$ user secret keys with constant size each; thus the size from this part is $O(\log n)$. Taking $n = \text{poly}(\ell)$ into consideration, each share size is bounded by

$$|C|\Big(\frac{1}{t_1} + \ldots + \frac{1}{t_\ell}\Big) + O(\log \ell).$$

Note that our ciphertexts have 2 more group elements than the plaintext, as we consider the case that ℓ is large, this little overhead is subsumed by the dominating terms. We conclude that our share size is $O[m \cdot (\frac{1}{t_1} + \ldots + \frac{1}{t_\ell}) + \log \ell]$. Taking a $(1, 2, \ldots, n)$-MSSS as example, it has a total size at only $O(\log \ell \cdot m)$.

Correctness: This is very similar to the correctness of the generic construction, with the only difference being that the combining keys are dispersed; however, as long as there are enough shares, the combining key can be reconstructed trivially from the IDS property.

Security: The only difference with the generic construction is that the combining keys are also dispersed here. Note that, they are given out directly in the generic construction, thus the security is not influenced. Combining Theorem 2 and Theorem 1, we conclude the section with the following theorem:

Theorem 3. Suppose λ is the security parameter, under the MSE-DH assumption, there exists a computational secure $(t_1, \ldots, t_\ell)$-multi-secret sharing scheme distributing ℓ secrets with length m each, having share size at $O(m \cdot (1/(t_1 + \ldots + 1/t_n)) + \log n\lambda)$.

4.4 An MPC Protocol with $O(\log n)$ Private States

A multi-party computation (MPC) protocol [48, 92] allows a set of parties to securely compute a function f in a distributed way while ensuring the privacy of the users' input and the correctness of the output, in the presence of dishonest users. Besides these basic *correctness* and *privacy* requirements, *robustness* (the honest parties learn their output even some parties play maliciously), *fairness* (if the honest parties do not learn their output, then the corrupted parties also cannot learn their output) are

also demanding properties for MPC protocols. Goldreich et al. [48] constructed the first general MPC protocols that can be either secure in the passive model even when there is only one honest user, or secure in the active model under the condition of honest majority. After this seminal work, most of the works on MPC focus on these two settings individually.

In [61], Ishai et al. aimed to achieve the best of both and combined the two protocols into one that enjoys full security (including all properties listed above) in the presence of any number of passive corruptions and a minority of active corruptions. Briefly, given the function f and the inputs of n parties $x_1, ..., x_n$, the protocol first computes $y = f(x_1, ..., x_n)$. However, the protocol does not directly output y. Instead, using a $(n/2, n)$-secret sharing scheme, the protocol outputs a sharing of y to each party. However, when there is no honest majority, and the adversary aborts during the reconstruction phase (at a moment when she might already has learnt y), the *fairness* property can not be guaranteed.

Hirt et al. [57] circumvented the above restriction and provide a a dynamic tradeoff between active and passive corruption, and their protocol has *fairness* even with dishonest majority (although the *robustness* still requires a honest majority). They introduce a new primitive as building block, called gradual verifiable secret sharing (VSS), which enables the protocol to release the secret gradually in the reconstruction phase. In this way, when the honest parties abort the protocol, the adversary could only have learned some shares of the secret.

To be more specific, a gradual VSS is a secret sharing scheme in which instead of the secret s, the random secrets $s_1, \ldots, s_n$ are shared among users where $s_1 + \ldots + s_n = s$ and each s_i is shared with threshold i. [3] In reconstruction protocol, the shares are

[3]The shares are the outputs of a general MPC protocol by running [48] in the first phase. Note

released one by one in decreasing order of thresholds. Briefly, for each $i \in [n]$, each user p_j first opens the commitment of the share s_{ij} via broadcast. If at least $(i+1)$ users correctly opened the commitments to the corresponding shares, then each user can locally calculate the secret s_i. Otherwise, the protocol is aborted, and each user outputs a set of corrupted users that did not correctly open the commitments. If there is no abort, each user will get the secret $s = s_1 + \ldots + s_n$ at the end of the protocol.

Remark that an abort at stage i prevents the adversary from learning $s_i, \ldots, s_1$ and thus the secret s. From this fact, the gradual VSS enables the protocol to preserve as much secrecy as possible. Given a function f and a set of inputs $(x_1, ..., x_n)$, the protocol given by [57] first computes $y = f(x_1, ..., x_n)$. It first splits y into $(n-1)$ secrets y_i. Similar to [61], instead of y, it just shares y_i among users by employing gradual secret sharing. If the users realize active corruption, they abort the protocol and output the set of corrupted users. They repeat the protocol with the new set of users that does not include the corrupted users in the previous execution. This protocol overcomes the drawbacks that the Ishai's protocol possess that we mentioned above.

The gradual VSS can be seen as a $(1, 2, \ldots, n-1)$-MSSS that reconstructs the secrets one by one in a decreasing order of the thresholds. However, to implement gradual VSS they use $(n-1)$ independent Shamir secret sharing schemes for each part y_i. Hence, the first phase of the protocol results in linear size shares for users (which they have store as private states, and the size grows fast when number of players is large) in the terms of the number users. However, our construction given in section

that even though the GMW protocol does not provide strong enough robustness or fairness, however, malicious parties do not gain any advantage aborting in this phase as they only see shares.

5.2 can be used as gradual secret sharing to distribute the values $s_1, \ldots, s_{n-1}$ where $\sum_{i=1}^{n-1} s_i = y = f(x_1, ..., x_n)$. Similarly, the protocol first runs the GMW protocol [48], computes the output value $s_1, \ldots, s_{n-1}$, and then all parties reconstruct the real secret gradually.

The only task left after replacing the "trivial" secret sharing scheme with our efficient MSSS is that we have to make it verifiable. In other words, the users should be convinced that the shares output by the protocol are valid. Besides, they have to prove the validity of their shares when they pool them to get the secrets s_i. Fortunately, it is not hard to turn our construction to a verifiable one. In brief, in the **Share** protocol, each user receives a secret key, ciphertext shares and combining key shares generated by the IDS. For the verifiability, the dealer has to prove that the secret key is in the form $g^{P_i(\gamma)}$ which can be done easily by checking some bilinear relation. The validity of the shares can be inherited from a verifiable IDS [22] (or even a general zero-knowledge proof can be carried out). While in the **Reconstruct** protocol, each user proves in zero-knowledge that the decryption share is in the form of $\sigma = e(g, h)^{k\alpha \cdot A(\gamma) \cdot P_i(\gamma)}$ which could be done efficiently (e.g., σ-protocol type of ZK proof [84]). At last, each user proves the validity of the shares of the ciphertexts and the combining keys. Note that for verifiability, some extra communication is necessary, however, they are not belong to the private states that each user has to store. For the details of each underlying tools, we refer to the references [22, 40, 84].

Chapter 5

Graded Signatures

In a petition system, a group of participants would like to send a formal request to an organization via a representative (petitioner) that helps them to express their opinions about an issue. There are several important criteria that a petition system has to satisfy: (1) the number of participants supporting the petition should be indicated; (2) the participants may prefer to remain anonymous in many scenarios, e.g., when these relate to political or religious issues; (3) the petitioner should not be able to make a false claim that the claimed number of participants is more than their actual number, e.g., duplicate participants should be removed without revealing any identities etc.

To address the above problem, we introduce a new primitive, which we call graded signature[1], that is applicable to an efficient privacy-preserving digital petition system. In a graded signature scheme a user collects signatures from registered signers in a PKI. The primitive enables the consolidation of an arbitrary number of signatures

[1]It is actually quite surprising that many seemingly related notions exist, however none of them satisfy all the natural requirements; we elaborate more on this below.

(say l) originating from a subset of l distinct signers on the same message m. The resulting signature object, $\sigma^{(l)}$, convinces the verifier that at least l signers indeed signed on m without revealing the identities of signers. We call l, the grade of the signature. Note that l can range from 1 to n, where n is the total number of currently registered signers in the PKI. There is no need to pre-determine the value of l before the signature collecting procedure.

There exist many variants of PKI oriented signatures that provide anonymity, e.g., ring signatures [81] is a prominent example. Moreover there are aggregate signatures [18] and threshold signatures [34, 86] which provide a form of a consolidation operation aimed at combining signatures into a single object. In some sense, a graded signature is a new primitive that brings together these lines of work. We will carefully compare our graded signature with existing related primitives below.

In a ring signature [81], the signer can anonymously sign a message on behalf of a group formed in an ad-hoc manner. Using our terminology, every ring signature will have a *fixed grade* 1, i.e., one of the signers signed the message. To form a graded signature with the grade k, for instance, the combiner can collect k ring signature on the message from k different signers. However, a regular ring signature scheme does not enable the receiver to check if there are two signatures produces by the same signer, which we need in graded signatures. To this aim, the notion of linkable ring signature [28, 69], that enables one to detect whether two signatures were generated by the same signer, seems sufficient to get a graded signature scheme at first glance. However, even if we use short ring signatures [3, 88], it still results in quadratic verification time since the verifier should check every pair of signatures. On the other hand, a (t, n)-threshold ring signature scheme [21] will convince the verifier that t signers agree on the message without leaking their identities. Similar to ring

signatures, any (t, n)-threshold ring signature will have a *fixed grade* t. While we also require anonymity in a graded signature in a similar sense, in contrast to these previous primitives, our graded signature should enable one to produce a signature, with an arbitrary grade, solely depending on how many signers agree to sign on the message; furthermore our constructions can even allow the grade to be upgraded if the user can get more signatures from additional signers on the message.

Regarding our second application of delegation of signing rights, one may think of proxy signature [14,72], in which a proxy can sign documents on behalf of the delegator if it is granted the signing rights from the owner by running a delegation protocol. Also, other variants of proxy signatures exist, e.g., anonymous proxy signature [42] provides anonymity for the intermediate proxies if there is a chain of delegatees; and threshold proxy signature [93] in which one key owner delegates his signing rights to a bunch of proxies, but only when the total number of proxies is above the threshold, a valid proxy signature can be produced. The notion of functional signatures was also studied in [20]. It enables the key owner to delegate the signing rights according to a fine-grained policy f, such that the delegatee can only sign messages in the range of f. Our notion of graded signature is different than the above in the sense that there are multiple key owners to delegate their signing rights to one "proxy", so that the "proxy signature" can be verified according to the number of delegators (its grade) without leaking the delegators' identities.kot

In a threshold signature scheme [34, 86] and its distributive variants [31, 77], when the number of signers is below the threshold, they can not jointly produce a signature that convinces the verifier. However, if the signers are above the threshold, the signatures will look the same to the verifier. Although the signers may be allowed to change [45], normally the value of the threshold needs to be fixed during the system

setup. Furthermore, they either require a fully trusted dealer to distribute signing keys, or when the number of signers is bigger than the threshold, they can recover all the secret key data. In contrast to that, in a graded signature, each consolidated signature is assigned a grade – the number of signers, and this number is not pre-determined, and it can vary from 1 to the total number of the signers which is n. Also, it can be deployed in a standard PKI setting without a trusted setup. No collusion of signers is able to produce a signature with grade larger than the size of the collusion.

The closest to our work is the notion of signature of reputation [9], which focuses only on the application of reputation systems and allows a user, as the combiner in our scheme, to consolidate all the upvotes for him as his reputation. Their construction is built on a general framework of NIZK proof systems that the user commits to each upvote and prove in zero-knowledge that each of the commitment contains a valid upvote. Note that a straigtforward application of such general framework would yield a signature of reputation with size that grows at least quadratically to the number of votes (even with the most efficient NIZK proof technique). The user has to provide a NIZK proof for each pair of commitments that they are from different identities. They resolve this problem via a clever use of the "linkability" of each commitment that the same randomness is used across all commitments of the votes, and each vote is essentially a unique signature. The verifier thus can check that each pair of commitments contain different votes which must come from different identities. However, this trick inherently incurs a quadratic (to the number of votes–grade in our terminology) verification time. Instead of only focusing on the application of reputation systems, our graded signature schemes aim at broader applications and we consider the notion as a more fundamental cryptographic primitive. Furthermore, since a signature might be verified many times, verification time is considered to

be one of the most important efficiency metrics. Moreover, we want to remove the restriction that only unique signature schemes can be used for graded signatures. We propose a new way of using the general commit and prove framework in our construction that brings down the verification time to linear while still keeps the signature size linear to the grade, for a broader class of signature schemes.

Another closely related work is graded encryption [65], which is a generalization of identity based encryption (IBE). The primitive enables the user to sequentially upgrade the level of his key so that the secret key of an identity with level k can decrypt all the ciphertexts sent for the identity with level $k' \leq k$. Since IBE implies a signature scheme a graded encryption scheme also implies a graded signature scheme in the sense that the consolidation has to happen in a sequential fashion. While our graded signature scheme does not have this restriction, the signatures can be collected in an arbitrary order from signers. It would be an interesting open question to consider graded encryption in the setting that the upgrading procedure is flexible like in graded signatures, i.e., only depending on how many secret keys received.

Besides those privacy preserving signature schemes, multi-signatures [7], and more generally, aggregate signatures [18] provide mechanisms for one to compactly represent signatures from different parties, some recent work [58] even shows that one may aggregate any type of signature from obfuscation techniques [58, 83]. However, the identities (public keys) of the signers will have to be explicitly given out for the verification. Contrary to that, a graded signature will keep the identities hidden, while reveal to the verifier only the grade of the signature. One may wonder whether adding some kind of anonymity to the signer to aggregate signatures will give us a graded signature. Specifically, if we have a trusted registration authority to issue certificates for public keys in a way that the identities are not revealed, the anonymity of the signers

will be achieved. However, graded signature schemes also require the distinctness of the signers to be validated. Besides, according to the application scenarios e.g., that of an anonymous petition, the definition of anonymity has to be very strong so that even the registration authority (which might be *the* adversary in some settings) is allowed to be corrupted. Actually this anonymity requirement is a crucial difference between aggregate signatures and graded signatures that makes them incomparable. On one hand, there is no clear mechanism from aggregate signatures that can provide us strong anonymity together with the proof of distinctness of signers; on the other hand, our strong anonymity precludes the possibility for the verifier to identify the exact source of the signature.

Our Results.

We first introduce formal definitions for graded signatures, including their correctness and security properties: unforgeability and anonymity. Every signer has his own key pair that is certified by the certificate authority. For correctness, when a signature is consolidated from ℓ different signatures, the consolidator should be able to convince the verifier that the signature is of grade ℓ' as long as $\ell \geq \ell'$. This allows us to define unforgeability focusing only at the attack scenario when the adversary produces a signature with grade one more than she is supposed to be able to produce. Regarding anonymity, we define it in a very strong sense: even if all parties, including the signers and the certification authority, are corrupted the consolidated signature should not leak the set of signers whose signatures were included in the consolidation process.

We provide an efficient construction for graded signatures which achieves a constant verification and secret key size while both the graded signature size and the

verification time are linear in the grade of the signature. This construction follows a "commit and prove" approach. Note that simply committing the signatures and showing that they originate from certified signers is insufficient: this is subject to a trivial attack where the consolidator uses the same signature over and over to increase the grade. In order to prevent this attack, an assurance of signature *distinctness* should be included in the proof that, if straightforwardly implemented, leads to a quadratic size or verification overhead. We go around this by introducing an order among signer public keys, and design the protocol in a way that it is compatible with the recent results of very efficient range proofs that were developed in [26].Note that since each signing key is independent, the verification key of each signer who contributes to the graded signature should be somehow involved in the signature object generation in order for the verification of the consolidated signature to take place correctly. Thus, if we view graded signature with grade l as a "proof of knowledge" of l signatures it follows that the length of the underlying consolidated signature must be at least linear in l since it is supposed to carry information for l independent originating signers. Besides, this proof should also include an argument which shows that l *distinct* certified signers were involved in its construction.

5.1 Definitions and Security Modeling

In a graded signature scheme, there is a set of signers who register their public key with a certification authority, as in a traditional PKI setting, and there is a procedure which enables a privacy preserving signature combining functionality. Specifically, from several signatures on a message m originating from different signers, one can produce

a "signature object" which convinces any verifier that at least "l distinct signers" signed on the message m without leaking the identity of any of them (beyond that they are members of the PKI of course). The grade l can vary from 1 to n where n is the total number of the registered signers in the system. For the ease of presentation, we differentiate the real grade ℓ which is the actual number of signatures used to consolidate the graded signature and the claimed grade ℓ' which is sent together with the graded signature for verification. Verfication algorithm will accept if $\ell' \leq \ell$. The detailed definition of a graded signature is as follows:

- **Setup**: This algorithm takes the security parameter as an input, and outputs a master key pair (gsk, gpk).

- **Register**: This algorithm takes the master secret key gsk, a signer verification key vk_i as inputs and outputs a certificate $cert_i = Sign(gsk, vk_i||i)$ for the registered signer. The index $i \in \{1, \ldots, n\}$ corresponds to a unique signer.

- **Sign**: This algorithm takes a key pair (sk_i, vk_i) and a message m as inputs, and outputs a signature σ_i on m.

- **Combine**: This algorithm takes as inputs the global public parameters gpk, a message m and a set of signatures $\{\sigma_{i_1}, \ldots, \sigma_{i_l}\}$ on m from different signers and a set of verification keys $\{vk_{i_1}, \ldots, vk_{i_l}\}$ and the corresponding certifications $\{cert_{i_1}, ..., cert_{i_l}\}$. It outputs a "consolidated" signature $\sigma^{(\ell)}$ and its real grade ℓ.

- **Verify**: This algorithm inputs the global public gpk, a message signature pair $(m, \sigma^{(\ell)})$ and the claimed grade ℓ' of the signatur, and outputs 0 or 1.

5.1.1 Security of Graded Signatures

The *Correctness* of a graded signature scheme requires that if ℓ valid signatures under ℓ different certified verification keys are used to produce the graded signature, then as long as the claimed grade is no bigger than ℓ, the verification should always output 1, i.e., if $(m, \sigma^{(\ell)}) = \textbf{Combine}(m, \{(vk_i, cert_i, \sigma_i)\}_{i=i_1,\ldots,i_l})$, and for each i, (m, σ_i) is a valid message-signature pair under vk_i, and $(vk_i, cert_i)$ is valid under gpk, then $\textbf{Verify}(\ell', m, \sigma, gpk)$=1 as long as $\ell' \leq \ell$.

There are two major security concerns in a graded signature, unforgeability and anonymity. Unforgeability in this setting means one can not produce a graded signature with a higher grade ($\geq \ell$) than that she is supposed to be capable of, i.e., she may register new users, corrupt existing users, and receive some signatures on a target message, but the numbers add up to at most $\ell - 1$. For anonymity, we require it in a very strong sense that any two graded signatures with a same grade will look indistinguishable (even to the CA and the signers who contribute one of the signatures).

Unforgeability of Graded Signatures: In order to capture all the possible attacks that the adversary $\mathcal{A}$ may try, we make explicit all kinds queries including registration queries which ask the CA to certify some public keys provided by $\mathcal{A}$, the corruption queries which enables $\mathcal{A}$ to learn the secret key of known, certified public keys, and the signature query for uncorrupted public keys. Consider the following game between an adversary $\mathcal{A}$ and a challenger C.

- $\mathcal{A}$ receives the master public key gpk.
- $\mathcal{A}$ is allowed to make registration queries, and gets certifications for the public keys that are generated by $\mathcal{A}$.

- $\mathcal{A}$ is also allowed to make corrupt queries, and gets secret keys for some existing certifications. (Note that all existing certifications and public keys together with the corresponding indices are available to the adversary)
- $\mathcal{A}$ also adaptively chooses messages to ask C for signing queries from signers that are not queried for the secret key or the certification, and receives the corresponding signatures on those messages.
- $\mathcal{A}$ outputs a message m^* and signature with grade l.

Definition 2. Let $Adv_{GS}^{\mathcal{A}}$ be the advantage of $\mathcal{A}$ in the game under the condition that $\mathcal{A}$ has asked at most $l-1 = q_1 + q_2 + q_3$ queries where q_1 is the total number of the secret key queries, q_2 is the total number of the certification queries, and q_3 is the total number of signature queries for m^*. We say the graded signature is existentially unforgeable under adaptive corruption attack if $Adv_{GS}^{\mathcal{A}} \leq negl(\lambda)$.

Remark that in our definition of unforgeability, we did not explicitly consider the attack that the adversary outputs a graded signature with $\ell + t$ for $t > 0$. However, from our definition of correctness, it is straightforward that if adversary is able to do so, she will be also capable of amounting an effective attack on our definition directly, as a forged signature with grade $\ell + t$ is also a forged signature with grade ℓ. We may also consider weaker models such as selective corruption, and we omit the discussion of details of these weaker variants.

Anonymity of Graded Signatures: We require a strong type of anonymity for a graded signature: two graded signatures can not be distinguished with respect to any characteristic except their grade. In the anonymity definition, even the certification authority and signers will not be able to link two graded signatures for an adversarially

chosen message with a same grade. Consider the following game between an adversary $\mathcal{A}$ and a challenger C.

- $\mathcal{A}$ receives the master public key gpk.
- $\mathcal{A}$ makes queries for the secret keys of signers. Note that the adversary here is allowed to corrupt all signers, even the certification authority.
- $\mathcal{A}$ also selects a grade l, a message m, and two sets of signers S_0, S_1 with size l such that $S_0 \neq S_1$. The adversary then produces two sets of tuples $D_0 = \{cert_i, \sigma_i, vk_i\}$ and $D_1 = \{cert_j, \sigma_j, vk_j\}$ where $i \in S_0$ and $j \in S_1$. Thus, $\mathcal{A}$ sends all sets S_0, S_1, D_0, D_1 and message m together with l to C.[2]
- The challenger C randomly flips a coin $b \in \{0, 1\}$, and sends $\mathcal{A}$ a graded signature $\sigma^{(\ell)}$ with grade l which is produced from l signatures on m from the set D_b.
- Finally, $\mathcal{A}$ output a guess b'.

Definition 3. We say a graded signature is fully anonymous if the probability of guessing the bit correctly is negligibly close to $\frac{1}{2}$, i.e., $|\Pr[b = b'] - \frac{1}{2}| \leq \epsilon$, where ϵ is a negligible function.

5.2 Graded Signatures with Linear Signature Size and Verification Time

In this section, we present an existentially unforgeable graded signature scheme with both linear in the grade verification time and signature size. The construction relies

[2]In order to simplify the game definition, we assume the sets S_0, S_1 differ only by one index, i.e., $S_0 \setminus S_1 = i_0$ and $S_1 \setminus S_0 = i_1$.

on involved mechanisms that are compatible with a constant size NIZK range proof together with a constant size NIZK proof of consistency of committed verification and signatures.

In order to motivate our construction recall the following generic solution for a graded signature: the user runs the aggregation algorithm of an aggregate signature scheme (or multi-signature with non-interactive signing) on input $vk_{i_1}, \ldots, vk_{i_t}, m, \sigma$, and commits to all the verification keys, and produces a non-interactive zero-knowledge proof for the following statements: 1. σ is an aggregate signature on m under the committed verification keys; 2. all committed verification keys are certified; 3. each of the committed verification keys are different. The straightforward way of proving the third condition in zero-knowledge would be to prove that the verification-keys are pairwise different. Even with the most efficient NIZK proof of inequality, this step brings a cost at least quadratic in the grade of the signature (the number of signer public keys) that we want to avoid. We may use SNARK [8]to construct efficient graded signature schemes as the final proof size could be as short as $poly(\lambda)$ where λ is the security parameter. However, we aim to get an efficient graded signature without applying knowledge assumptions, thus we will focus on using standard building blocks as Groth-Sahai proofs [53] below.

Besides, designing a linear size signature from standard assumptions was also the main technical work of signature of reputation [9]. Unfortunately their technique inherently relies on certain kind of "linkability" among commitments and "uniqueness" of the signature scheme. They incur quadratic verification time and restrict the class of signature schemes that can be used to produce a graded signature.

We go around these problems by introducing a new technique that we assign an index from $\{1, ..., n\}$ as a part of public key of the signer where n is the maximum level.

We then utilize an efficient non-interactive range proof so that we can sort the indices and sequentially prove a "larger than" statement to show that indices from which the graded signature is produced are different. In this way we can bring down the complexity from (at least) quadratic to linear. Specifically, when a signer registers his verification key, the CA will choose an index for him and sign the index together with his verification key to produce the certificate for that signer. After collecting signatures $(m, vk_{i_1}, \sigma_{i_1}, cert_{i_1}), \ldots, (m, vk_{i_t}, \sigma_{i_t}, cert_{i_t})$, the **Combine** algorithm commits to all the verification keys, all the certificates, and all the corresponding signatures. Then, the **Combine** algorithm will produce a proof that each committed signature is valid under the corresponding committed verification key; second, a proof that each certificate is valid under the public key of the certification authority; third, the algorithm will sort the indices of the verification-keys in a decreasing order and establish that each index belongs to range $[1, n]$. Due to the additive homomorphic property of the commitment scheme we use, the **Combine** algorithm will be also capable to produce a proof that $Com(i_j - i_{j-1})$ is a commitment to an integer which also falls in range $[1, n]$. So it follows that this value is bigger than 0, and hence the difference of any two neighboring indices is strict. In this fashion the algorithm will establish a proof showing that there are l valid signatures from l different certified signers on m. This completes the high level overview of the construction. What remains is how to get a constant size NIZK proof for each of the above statements. Thanks to the flexibility of this construction methodology we can choose any appropriate signature scheme as long as it can be paired with efficient NIZK proofs. We instantiate the scheme using automorphic signatures [1] together with a Groth-Sahai proof of validity of committed signatures [54], and also an efficient range proof of committed values. In this way we can see that verification only has to do a sequential scanning instead

of pair-wise comparison as in [9]; furthermore, the signarture size is still linear in the grade as each component only cost a constant number of group elements.

Suppose we have two signature schemes Sig, Sig', an additively homomorphic commitment scheme Com. The scheme is formally presented as follows:

- **Setup**: The algorithm runs the key generation of Sig, and generates a key pair (msk, mpk). It also generates global parameters $param$ including the CRS string for the commitment scheme and the NIZK proof system, and the total number n of allowed signers in the system. It outputs the global key pair (gsk, gpk) where $gsk = msk$, and $gpk = (mpk, param)$.

- **Register**: This is a protocol between signer and CA. Signer first runs the key generation of sig' to get his signing key pair (vk, sk), and submits vk to the CA. The CA first checks whether this signer is already registered, if not, he chooses an index i, runs the signing algorithm of Sig on (vk, i), and returns the signer $cert_i$, where $cert_i = Sig(msk, (vk, i))$.

- **Sign**: This algorithm receives as input a signer's secret key sk_i, a message m and runs the Sig' algorithm to get a signature σ_i on m, and it outputs σ_i, signer's index i and $cert_i$.

- **Combine**: This algorithm takes as inputs a message m, a sequence of signatures $(\sigma_{i_1}, \ldots, \sigma_{i_l})$ for the message m under $vk_{i_1}, \ldots, vk_{i_l}$ with the corresponding certificates $(cert_{i_1}, \ldots, cert_{i_l})$, from l different signers. It first checks the validity of the signatures and the certificates, and determines the grade l. Suppose the sequence is in a decreasing order according to the indices, i.e., $i_1 > i_2 \ldots > i_l$. It computes the commitments to all those values and gets $c^1_{i_j} = Com(\sigma_{i_j})$ for the

signatures, $c^2_{i_j} = Com(vk_{i_j})$ for the signers' verification keys, $c^3_{i_j} = Com(cert_{i_j})$ for the certificates, and $c^4_{i_j} = Com(i_j)$ for the signers' indices. Using the signatures as witnesses, it constructs $4l - 1$ NIZK proofs. For each $j \in \{1, \ldots, l\}$, the proof $\pi^1_{i_j}$ establishes that $c^1_{i_j}$ commits to a valid signature on m under the verification key contained in $c^2_{i_j}$; $\pi^2_{i_j}$ proves that $c^3_{i_j}$ commits to a valid signature under mpk on the message pair contained in $c^2_{i_j}$ and $c^4_{i_j}$; $\pi^3_{i_j}$ proves $c^4_{i_j}$ commits to a value which belongs to $\{1, \ldots, n\}$; $\pi^4_{i_j}$ proves that $c^4_{i_{j+1}}/c^4_{i_j} = Com(i_{j+1} - i_j)$ also commits to a value ranging in $\{1, \ldots, n\}$. It outputs the message m and signature object as $\{c^1_{i,j}, c^2_{i_j}, c^3_{i_j}, c^4_{i_j}, \pi^1_{i,j}, \pi^2_{i_j}, \pi^3_{i_j}, \pi^4_{i_j}\}_{j=1,\ldots,l}$, together with its grade l.

- **Verify**: The verifier takes global public key gpk, a message m, and a graded signature $\{c^1_{i,j}, c^2_{i_j}, c^3_{i_j}, c^4_{i_j}, \pi^1_{i,j}, \pi^2_{i_j}, \pi^3_{i_j}, \pi^4_{i_j}\}_{j=1,\ldots,l}$ with grade l as inputs, it first parses the signature, and for $j = 1, \ldots, l$, it checks the validity of the proofs $\pi^1_{i,j}, \pi^2_{i_j}, \pi^3_{i_j}$, and for $j = 1, \ldots, l-1$, it checks the validity of $\pi^4_{i,j}$; if all checks pass, it outputs 1, otherwise 0.

Correctness: The correctness of our scheme trivially follows the correctness of the signature schemes Sig and Sig', and the completeness of the NIZK proof systems. Briefly, if the user has ℓ signatures $(\sigma_{i_j}, vk_{i_j}, cert_{i,j})$ for a message m collected from different signers such that each σ_{i_j} is a valid signature on m under vk_{i_j}, and each $cert_{i_j}$ is valid signature on (pk_{i_j}, i_j) under mpk, and if the ℓ tuples $(\sigma_{i_j}, vk_{i_j}, cert_{i,j})$ are sorted in decreasing order, and all indices i_j and all $i_{j+1} - i_j$ are in the range $[1, n]$. Then from the completeness of NIZK proof systems, the Verify algorithm accepts the

signature $\sigma^{(\ell)}$ on m constructed as

$$\sigma^{(\ell)} = \{c^1_{i,j}, c^2_{i_j}, c^3_{i_j}, c^4_{i_j}, \pi^1_{i,j}, \pi^2_{i_j}, \pi^3_{i_j}, \pi^4_{i_j}\}_{j\in[\ell]} = Combine(m, (\sigma_{i_j}, vk_{i_j}, cert_{i_j})_{j\in[l]}).$$

Security Analysis: Security follows quite easily from the properties of the zero-knowledge proofs and the commitment schemes. For *unforgeability*, suppose the adversary only gets t signatures on a message m by corrupting signers or asking signing queries, and he is able to produce a signature on m with grade $t+1$. According to the soundness of the NIZK proof system, there must be $t+1$ valid signatures under $t+1$ different verification keys committed by the adversary. Note that because of the extractability property of the commitment scheme, at the beginning, the simulator can produce a simulated crs which contains an opening trapdoor for the commitment scheme, and thus the simulator can open these commitments to retrieve the $t+1$ tuple of signatures, verification keys, and certificates. If the verification keys are all certified by the the CA, then the adversary must have forged one new signature against an honest signer; alternatively, the adversary could have forged a certificate for an unregistered verification key. The simulator can examine these cases and break the unforgeability of either Sig' or Sig.

Regarding *anonymity*, suppose the adversary submits m, l, S_0, S_1, D_0, D_1 as the challenge. Suppose, for simplicity, that $S_0 \setminus S_1$ contains only one index i_0 and similarly $S_1 \setminus S_0$ contains only one index i_1. The simulator can use signatures $\sigma_{i_0}, \sigma_{i_1}$ on m under pk_{i_0}, pk_{i_1} to ask as a challenge in a plaintext indistinguishability game of the underlying commitment scheme; after receiving $Com(\sigma_{i_b})$, the simulator will create a graded signature by computing the commitments to all other signatures on m and simulate all the proofs (the latter part following from the zero-knowledge property).

In this way, the simulator can use the adversary's ability in breaking anonymity to break the hiding property of the commitment scheme in a straightforward fashion.

Theorem 4. The scheme is existentially unforgeable under adaptive corruption attacks if Sig, Sig' are unforgeable digital signatures, Com is a binding (extractable) commitment scheme, and the proof system is sound.

Proof: We show the security by a sequence of games. We start with the original game Game_0, and prove that a polynomial time attacker's advantage of distinguishing any successive games is negligible.

Game_0 :

- The simulator runs the key generation of Sig, and generates a key pair (msk, mpk). It also runs the key generation algorithm of Sig' to generate the signing key-verification pairs. Then for each verification key vk, it picks a random integer $i \in [n]$, generates the certification of the corresponding verification key using $MS.Sign$ algorithm on (vk, i), and forms a set S that contains all certifications and corresponding indices. Besides, it generates the global parameters $param$ including the crs strings for the commitment scheme and the NIZK proof system, and the total number n of allowed signers in the system. The simulator keeps gsk, and gives $gpk = (mpk, param)$ and S to the adversary.

- For each register query; the adversary $\mathcal{A}$ generates a fresh key pair $(sk, vk) \leftarrow S.Setup$, and gives vk to the simulator. The simulator selects a random integer i from $[n] - S$ as the index of vk (The challenger keeps a list T for registered indices. If $i \in T \cup S$, the simulator reselects it) and computes $cert_i = \text{MS.Sign}(msk, (vk_i, i))$. $\mathcal{C}$ sends $cert_i$ to $\mathcal{A}$ and writes $(i, vk, cert_i)$ to T.

- For each signing key query of an index $j \in S$; the simulator gives the corresponding signing key sk_j to the adversary. The simulator also keeps a list C for corrupted indices.

- For each signature query on the message m with index $k \in [n]-T$; the simulator computes $\sigma = \text{S.Sign}(sk_k, m)$, and gives it to $\mathcal{A}$.

- $\mathcal{A}$ submits a forgery $\sigma^{(\ell)}$ with grade $\ell > |C| + |T| + q$ for m^* where q is the number of signature queiries on m^*. If $\text{Verify}(\sigma^{(\ell)}, \ell, m^*, gpk) = 1$, the adversary wins the game.

Game$_1$: Same as Game$_0$, except we substitute Setup algorithm of the commitment scheme with Extractable Setup algorithm which generates the crs string of the commitment scheme together with the extraction key ek.

Game$_2$: Same as Game$_1$, except we require that for each $c^u_{i_j} = Com(crd, (X^u_{i_j}, \alpha))$ and associated proof $\pi^u_{i,j} \leftarrow Prove(crs, Ver^u_{i_j}, (X^u_{i_j}, \alpha))$ generated by the adversary in the challenge phase, $Ver(crd, E^u_{i_j}, c^u_{i_j}, \pi^u_{i,j}) = 1$ where $E^u_{i_j}$ is the corresponding verification equation.

Claim: Assuming the NIZK proof systems has two types of common reference strings (hiding and binding) which are computationally indistinguishable, for any PPT adversary $\mathcal{A}$,

$$|Adv^{(0)}_{\mathcal{A}} - Adv^{(1)}_{\mathcal{A}}| \leq negl(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference the advantages of the adversary between both games is non negligible, then we can construct a PPT algorithm $\mathcal{B}$ that use $\mathcal{A}$ to distinguish two types of CRS with non negligible advantage. □

Claim: Assuming the NIZK proof systems are sound , for any PPT $\mathcal{A}$,

$$|Adv_{\mathcal{A}}^{(1)} - Adv_{\mathcal{A}}^{(2)}| \leq negl(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference the advantages of $\mathcal{A}$ between both games is non negligible, then we will construct a PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the soundness of the proof systems.

$\mathcal{B}$ gets the crs of the commitments from the challenger of the NIZK proof system. It then computes $(msk, mpk) \leftarrow MS.Setup(\lambda)$ and gives the gsk, gpk to the adversary. $\mathcal{B}$ can simulate the corrupt queries, the registration queries, and the signature queries as in Game_1 and Game_2. The only difference between two games is that, the adversary can prove a false statement with non-negligible probability in Game_1. If the algorithm $\mathcal{B}$ is dealing with the proofs of false statements, then it corresponds to Game_1; otherwise it corresponds to Game_2. Thus, $\mathcal{B}$ can break the soundness of the underlying proof systems with non-negligible probability. □

Claim: Assuming Sig and Sig' are existentially unforgeable, and the commitment scheme is perfectly binding, for any PPT $\mathcal{A}$,

$$Adv_{\mathcal{A}}^{(2)} \leq negl(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference the advantages of $\mathcal{A}$ between both games is non negligible, then we will construct a PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the unforgeability of Sig or Sig'.

$\mathcal{B}$ gets mpk from the challenger of Sig and vk from the challenger of Sig', and requests a certification for vk. $\mathcal{B}$ also generates some signing key-verification key

pairs, and requests the certifications for those verification keys. It then forms the set S that contains all certifications and corresponding indices. Besides, $\mathcal{B}$ generates the global parameters $param$ that includes the extractable crs strings for the commitment scheme with the extraction key ek and the NIZK proofs system, and the total number n of the allowed signers in the system. $\mathcal{B}$ keeps ek, and gives $(mpk, params)$ to the adversary.

For each register query that the adversary makes, $\mathcal{B}$ gets the corresponding certification from the challenger of Sig; for each signing key query that the adversary makes, if the corresponding verification key is vk, $\mathcal{B}$ aborts, otherwise gives the corresponding signing to the adversary; for each signature query on a message m, if the adversary requests a signature for vk, $\mathcal{B}$ asks a signature on m from the challenger of Sig', otherwise produces the signature using the corresponding signing key.

When the adversary submits a valid forgery $\sigma^{(k)}$ on a message m^* with the grade ℓ, $\mathcal{B}$ extracts all tuples $\{(vk_i, cert_i, \sigma_i)\}_{i\in[\ell]}$ uniquely from $\sigma^{(k)}$ using ek since the commitment scheme is perfectly binding (extractable). Since the number of registration queries and the corruption queries add up to be less than ℓ, there should be one tuple $(vk_i, cert_i, \sigma_i)$ such that either σ_i is a valid forgery on m^* under vk_i, or $cert_i$ is a valid forgery on (vk_i, i) under mpk. If σ_i is a valid forgery on m^*, since the probability of $vk_i = vk$ is $1/|S|$, $\mathcal{B}$ can use this forgery to break the unforgeability of Sig. If $cert_i$ is a valid forgery on (vk_i, i), then $\mathcal{B}$ can use the pair to break the unforgeability of Sig'. This concludes the proof. □

Theorem 5. The scheme satisfies full anonymity, if Com is computationally hiding and the proof system is zero-knowledge.

Proof: We show the security by a sequence of hybrid experiments. We start with the

original experiment Game_0, and prove that any polynomial time attacker's advantage of distinguishing any successive experiments is negligible.

Game$_0$:

- The challenger runs the key generation of the scheme Sig, and generates a key pair (msk, mpk). It also generates global parameters $param$ including the crs strings for the commitment scheme and the NIZK proof system, and the total number n of allowed signers in the system. The challenger gives $gpk = (mpk, param)$ and gsk to the adversary.

- The adversary selects two sets S_1 and S_0 of indexes such that $|S_0| = |S_1| = k$, $S_1 \setminus S_0 = \{i_0\}$, and $S_0 \setminus S_1 = \{j_0\}$. It first runs $S.Setup$ algorithm to generate signing key-public key pair (sk, pk) for each index, then computes a certification $cert_i$ for each public key vk_i of the index i using gsk. The adversary also produces signatures σ_i on same message m under each public key pk_i. It finally gives the index k as the level, the message m, two sets of indexes S_0, S_1, and two sets of tuples $D_0 = \{cert_i, \sigma_i, pk_i\}$, $D_1 = \{cert_j.\sigma_j, pk_j\}$ where $i \in S_0$ and $j \in S_1$.

- The challenger sets $b = 0$, produces a graded signature $\sigma^{(k)}$ on m using the tuples D_b, and gives $\sigma^{(k)}$ to the adversary.

- The adversary gives a guess b' to the challenger, and wins the game if $b = b'$.

Game$_1$: Same as Game_0, except we substitute Setup algorithm of the commitment scheme with SimSetup algorithm which generates the simulable crs string of the commitment schemes and proofs.

Game$_2$: Same as Game$_1$, except that the challenger changes $(\pi^1_{i,0}, \pi^2_{i_0}, \pi^3_{i_0}, \pi^4_{i_0})$ of index i_0 from $\sigma^{(k)}$ with the simulated proofs $(\pi'^1_{i_0}, \pi'^2_{i_0}, \pi'^3_{i_0}, \pi'^4_{i_0})$.

Game$_3$: Same as Game$_2$, except that the challenger changes the commitments $(c^1_{i,0}, c^2_{i_0}, c^3_{i_0}, c^4_{i_0})$ of index $i_0 \in S_0$ from $\sigma^{(k)}$ with $(c^1_{j,0}, c^2_{j_0}, c^3_{j_0}, c^4_{j_0})$ of the index $j_0 \in S_1$.

Game$_4$: Same as Game$_3$, except that the challenger changes the simulated proofs $(\pi'^1_{i_0}, \pi'^2_{i_0}, \pi'^3_{i_0}, \pi'^4_{i_0})$ from $\sigma^{(k)}$ with the proofs $(\pi^1_{j_0}, \pi^2_{j_0}, \pi^3_{j_0}, \pi^4_{j_0})$. Thus, in the final game, the challenger generates the graded signature $\sigma^{(k)}$ using the tuples from the set D_1.

Claim: Assuming the proof systems are zero-knowledge, for any PPT $\mathcal{A}$,

$$|Adv^{(0)}_{\mathcal{A}} - Adv^{(1)}_{\mathcal{A}}| \leq negl(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference of the advantages of the adversary between both games is non negligible, then we can construct a PPT algorithm $\mathcal{B}$ that use $\mathcal{A}$ to break the zero knowledge property of the proof systems. □

Claim: Assuming the proof systems are zero-knowledge, for any PPT $\mathcal{A}$,

$$|Adv^{(1)}_{\mathcal{A}} - Adv^{(2)}_{\mathcal{A}}| \leq negl(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference of the advantages of the adversary between both games is non negligible, then we will construct a PPT algorithm $\mathcal{B}$ that use $\mathcal{A}$ to break the zero knowledge property of the proof systems.

$\mathcal{B}$ generates (gsk, gpk) and gives them to the adversary as in Game$_1$ and Game$_2$. After getting it, the simulator gives the challenge tuple $(vk_{i_0}, cert_{i_0}, \sigma_{i_0})$ to the chal-

lenger of the proof system, and gets the corresponding commitments com_{i_0} and proofs $\pi_{i_0}^{(b)}$. $\mathcal{B}$ then simulates all other commitments and proofs and gives the final signature to the adversary. If $b = 0$, then it corresponds to Game_1, otherwise it corresponds to Game_2. Thus, if the difference of the advantages of the adversary between both games is non negligible, then $\mathcal{B}$ can use $\mathcal{A}$ to break the zero knowledge of the proof system. $\square$

Claim: Assuming the commitment scheme is computationally hiding, for any PPT adversary $\mathcal{A}$,

$$|Adv_{\mathcal{A}}^{(2)} - Adv_{\mathcal{A}}^{(3)}| \leq negl(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference of the advantages of the adversary between both games is non negligible, then we will construct a PPT algorithm $\mathcal{B}$ that use $\mathcal{A}$ to break hiding property of the commitment scheme.

$\mathcal{B}$ generates (gsk, gpk) and gives them to the adversary as in Game_2 and Game_3. After getting it, the simulator gives the tuples $(vk_{i_0}, cert_{i_0}, \sigma_{i_0})$ and $(vk_{j_0}, cert_{j_0}, \sigma_{j_0})$ to the challenger of the commitment scheme, and gets the challenge commitments $(c_b^1, c_b^2, c_b^3, c_b^4)$. $\mathcal{B}$ also simulates all other commitments and corresponding proofs, and gives the final signature to the adversary. If $b = i_0$, then it corresponds to Game_2, otherwise it corresponds to Game_3. Thus, if the difference of the advantages of the adversary between both games is non negligible, then $\mathcal{B}$ can use $\mathcal{A}$ to break the hiding property of the commitment scheme. $\square$

Claim: Assuming the proof systems are zero-knowledge, for any PPT $\mathcal{A}$,

$$|Adv_{\mathcal{A}}^{(3)} - Adv_{\mathcal{A}}^{(4)}| \leq negl(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference of the advantages of the adversary between both games is non negligible, then we will construct a PPT algorithm $\mathcal{B}$ that use $\mathcal{A}$ to break the zero knowledge property of the proof systems.

$\mathcal{B}$ generates (gsk, gpk) and gives them to the adversary as in Game_3 and Game_4. After getting it, the simulator gives the challenge tuple $(vk_{j_0}, cert_{j_0}, \sigma_{j_0})$ to the challenger of the proof system, and gets the corresponding commitments com_{j_0} and proofs $\pi_{j_0}^{(b)}$. $\mathcal{B}$ then simulates all other commitments and proofs and gives the final signature to the adversary. If $b = 0$, then it corresponds to Game_3, otherwise it corresponds to Game_4. Thus, if the difference of the advantages of the adversary between both games is non negligible, then $\mathcal{B}$ can use $\mathcal{A}$ to break the zero knowledge of the proof system.

In conclusion, since any PPT attacker's advantage of distinguishing any successive games is negligible, the adversary cannot distinguish two graded signatures with the same grade. Hence, the scheme is fully anonymous. □

5.2.1 An Efficient Instantiation

In order to get a graded signature with size linear in the grade, we need to make all the NIZK proofs $\pi_{ij}^1, \pi_{ij}^2, \pi_{ij}^3, \pi_{ij}^4$ to be constant size. One natural approach, that also yields a standard model construction, is to instantiate the scheme with signature schemes which are compatible with the Groth-Sahai proof system [54]. Note that a structure preserving signature or automorphic signature [1] satisfies exactly our needs – both the verification key and the signature belong to the same group, and the verification are conjunctions of pairing product equations; furthermore, this signature

scheme allows signing on a pair of messages as well. For Sig, the CA needs to sign on pk and index i; we instantiate this with an automorphic signature on (g^x, g^i), where $pk = g^x$; for Sig', in order to sign a message $m \in Z_p$, we instantiate the algorithm via the same signature scheme operating on a single group element equal to g^m. It is straightforward to obtain constant size proofs realizing π_i^1, π_i^2 by applying the Groth-Sahai framework.

For π_i^3, π_i^4, we use the constant size non-interactive range proof for range $[0, H]$ proposed in [26] [3]. First, we apply the range proof for the range $[1, n]$ in order to establish the "larger than" statement. Relying on the additive homomorphic property of the commitment scheme, we can do a straightforward "shift" in the protocol of [26], in order to prove $x \in [1, n]$, where x is committed in $Com(x)$. Specifically the prover executes the proof with respect to the commitment $\psi = Com(x)/Com(1; 0)$ where $Com(x; r)$ denotes the commitment on x with randomness r, thus establishing that $x - 1 \in [0, H]$. With this construction at hand, it follows that the proofs π_i^3, π_i^4 are also constant size.

Now the only problem left is to show the index committed for π_i^1, π_i^2 is consistent with the value committed for π_i^3, π_i^4. We observe that the commitment schemes used in the range proof include a BBS encryption type of commitment, which is compatible with Groth-Sahai proof system and this proof can be constructed easily. Specifically, the NIZK proof establishes that the two commitments c_1, c_2 belong to the language:

$$L = \{(c_1, c_2) | \exists x, r_1, r_2, s_1, s_2, s.t, c_1 = (g^{x+r_1+r_2}, f^{r_1}, h^{r_2}) \wedge c_2 = (g^x u_1^{s_1} u_2^{s_2})\},$$

[3] Using different instantiations of parameters, they obtain suitable communication and verification complexity for different scenarios. In our case, adding CRS with $O(log^{1+\epsilon} n)$-length to the public parameters will be enough to achieve constant size range proof and verification time.

where g, f, h, u_1, u_2 are all contained in CRS.

5.3 Graded Signatures Supporting Revocation

Since our notion of graded signature is directly built upon the PKI, it would be nice if we can support certificate revocation as well due to the same reasons as in the regular PKI setting, e.g., some signing key might get compromised. A common method for revocation in the PKI setting is that the CA publishes a revocation list that maintains all the revoked certificates, and every user can check it.

In our construction of the graded signature scheme, in order to guarantee that the signatures are all from the valid signers and their privacy is preserved, we have one important step that the user commits to the certificates and the public keys and proves that the public keys are certified, i.e., the certificates contained in the commitments are valid signatures under the master public key of the CA. We can see that in principle, it would not be very difficult to extend our construction to support revocation as we can simply let the user to add one more proof that the certificates committed are *not* in the public revocation list. [4]. The challenging task is that how we can maintain the signature size still to be linear in the grade, which means we need to keep each non-membership proof to be constant! [5]

Fortunately, Blazy et al [11] propose an efficient NIZK proof system to prove an exclusive statement, i.e., the statement does not belong to a language L. We can instantiate their proof system to prove that a committed value does not belong to a

[4]Instead of certifications, it would be enough to keep only the indices of the revoked signers in the revocation list.

[5]A straightforward way to show that the committed value does not equal to any of the set element is highly inefficient due to the inequality proof and the AND proof.

given set S. The main idea of their technique is that the user first generates a "proof" $\tilde{\pi}$ showing that the statement belongs to L, and it can not pass the verification (as he does not have the witness), then he proves using another $\tilde{\pi}'$ that $\tilde{\pi}$ is generated honestly, i.e., it is indeed computed following the regular prover algorithm. In this way, $\tilde{\pi}, \tilde{\pi}'$ together convince the verifier about the negation, as if not, the prover can not generate $\tilde{\pi}, \tilde{\pi}'$ simultaneously. For details of the technique, we refer to [11]. Now to instantiate the non-membership proof, we can start with the membership proof we use [11] to generate $\tilde{\pi}$ which is constant size and we then prove each component of $\tilde{\pi}$ is generated honestly. Since [11] is compatible with Groth-Sahai [53], the validity of the components can be again proven efficiently using the Groth-Sahai proof. Thus, we can conclude that we can extend our graded signature to support certificate revocation by adding the above non-membership proof for each committed certificates. Furthermore, each pair of such non-membership proof is with constant size, thus the total signature size is still linear in the number of grade.

5.4 Applications

The new primitive can be useful in a number of applications that we discuss below.

Anonymous Petitions: In an anonymous petition, the petitioner aims at convincing an organization that a certain number of people have a consensus on one issue, and it is desired that the identity of each participant remains hidden. Our graded signature immediately solves this problem. Suppose every valid voter has a registered public key, and the one who initiates a petition on a message m, tries to get as much support (signatures) on m as she can. At the end, she consolidates all the signatures

into one, and presents to the organization the message, the graded signature and the corresponding grade l. The privacy of all signers will be preserved, and the grade precisely reflects how many signatures the consolidator collected. The organization can verify that indeed l different signers are needed to produce the l-grade signature using the PKI parameters.

Anonymous Delegation of Signing Rights adhering to Threshold Policies: Consider an organization whose members are in a PKI and wish to authorize in anonymous fashion a certain individual to execute certain tasks without necessarily revealing their names. The authorization requires a certain quorum that, if reached, it should be universally accepted. For instance, suppose that the members of the board of trustees of a listed company would like to authorize the CEO to take certain decisions on behalf of them. Such authorization may require the agreement of the majority (or other suitable percentage) of the trustees. Using graded-signatures the CEO can obtain the signature of a suitable number of trustees on her public-key and then consolidate those to demonstrate the fact that a suitable number of trustees endorse her actions.

Graded Certificates for multi-CA PKI's: As a number of incidents have shown, certification authorities (CA) can be corrupted (e.g, see [63]) and in this way the security of critical Internet protocols such as TLS can be jeopardized. In a multi-CA setting a user may obtain certificates from multiple PKI's tying her identity to her public-key. Assuming the CA's themselves can be certified by an acceptable top-level CA, a user can form a "graded certificate" by consolidating her distinct certificates coming from different CA's into a single graded signature. The grade will reflect the number of certificates that the user has collected on her identity. Using graded

signatures it is thus possible to enable a certificate negotiation step between two communicating parties that (1) provides sufficient assurance on their identities (by requiring a minimum signature grade for both sides) and (2) maintains their privacy in terms of their CA choices as the anonymity of the graded signature reveals only the grade but not the individual entities that have provided certificates.

Chapter 6

Graded Encryption

In the game show "who wants to be a millionaire", participants earn money as rewards by answering a series of consecutive multiple-choice questions. The questions are distributed in several rounds, and the difficulty and the amount of money of the questions increase with the rounds. After correctly answering the question in each round, the contestants can either choose "walk away" with the current rewards or choose "continue" to go after a higher reward with the understanding that if they fail in the next round they will loose everything. The above description is a simplified version of the game, but it captures the essential idea of it.

In a live game that operates in stages with an increasing sequence of rewards, one can implement a scheme that the audience can audit and a trusted host can ensure the distribution of the questions and the reward. In a more general setting, in multi-stage computer games a player has to advance the game stage-by-stage, and receive a corresponding reward for advancing. The rewards could be monetary, but also virtual devices that make the player more competitive in the next stages so that

the player expects to "unlock" such device from some existing elements when they complete some tasks at a certain level. In a distributed version of all those games, managing multi-stage games for a large amount of players may be quite complicated, and especially in some computational intensive 3-D video games, it may be required that multiple servers are needed to handle the game management for different stages. Moreover, the servers can be targeted by attackers who want to pass the game or directly "steal" the rewards.

We propose a new identity-based cryptographic primitive which we call graded encryption. The new primitive enables us to design the reward distribution mechanism for such online games distributively in a way that the damage from the corruption of the servers can be reduced to as little as possible, and the task of managing the games and rewards becomes as simple as possible. In a graded encryption scheme, there is one central (mostly offline) authority and a number of sub-authorities holding master keys that correspond to different levels. As in identity-based encryption, a sender can encrypt a message using only the identity of the receiver (plus public parameters) but it may also specify a numerical grade i. Users may decrypt messages directed to their identity at grade i as long as they have executed a *key-upgrade* protocol with sub-authorities $1, \ldots, i$. We require a grade i ciphertext to be secure in a strong sense: as long as there is one sub-authority with index $j \leq i$ that is not corrupted, the plaintext should be hidden from any recipient that has not properly upgraded her identity.

Identity based encryption [17, 47, 90] could be possibly used to build a graded encryption generically, but a black-box construction incurs an overhead which is linear in the number of levels (more detailed discussion is in section 3.2). In an hierarchical IBE [59], a user with identity a_1 can derive secret key for user with identity $\langle a_1, a_2 \rangle$,

but not the other way around. One may think of using an HIBE to construct a GE, starting from an identity with a largest level $\langle a_1, \ldots, a_n \rangle$ and gradually improving his grade by obtaining keys for $\langle a_1, \ldots, a_{n-i} \rangle$. However, it is not obvious how to achieve this efficiently. We want to emphasize that in a GE scheme, there is no hierarchy among sub-authorities in the sense of HIBE, i.e., sub-authority of level i cannot decrypt ciphertexts of level $i - 1$.

In threshold public key encryption systems [4, 15, 24, 35], at least t parties jointly decrypt a ciphertext. The threshold of the system should be fixed during setup. In [33], an extension of threshold public key encryption to the dynamic setting was proposed so that any user can dynamically join the system, and the sender can dynamically set the threshold in each encryption. Once the adversary corrupts more than the threshold shareholders, the security collapses. In a graded encryption, on the other hand, even if all except one authority is corrupted, we can guarantee the security of the ciphertext which has a level greater than the index of the uncorrupted authority. Furthermore, the authorities in a GE do not communicate or can not decrypt messages, only provide assistance to the user to upgrade his decrypting ability. The threshold encryption schemes do not provide an "ordered" mechanism suitable for the type of sequential upgrading that we consider here is needed.

In multiple encryption [38, 74, 76], a message is encrypted several times with independent keys or even using different encryption algorithms to provide better security. As suggested by [37], the message m can be split into shares $m_1, ..., m_i$ and all shares can be encrypted using a public key pk_i associated to the authority i. A user is only able to decrypt the ciphertexts when they have all corresponding secret keys $sk_1, ..., sk_i$ taken from the associated authorities. This trivial solution has the secret key size and the ciphertext size linear in the grade of the ciphertext.

Ateniese and Hohenberg [2] showed how multi-use proxy re-signatures, which allow a proxy to convert a valid signature of a message from Alice into a valid signature of same message from Bob, can be elegantly used to prove that a certain path was taken in a graph (our third application). Our graded encryption scheme provides an alternative solution for this application. Our construction also applies to the applications that require encryption and as such they are out of the scope of the scenarios that proxy re-signatures are applicable.

Our Results.

We formalize a new cryptographic primitive as an extension of IBE that we call graded encryption (GE). In a graded encryption scheme, there is a central authority and a number of sub-authorities corresponding to fixed indices. GE enables a sender to specify a grade for a ciphertext depending on the importance of the plaintext. The receiver of the ciphertext can decrypt it only if he holds a secret key with the same grade (or higher grade depending on the implementation). Here the level of a secret key determines the number of the authorities the receiver should contact (we will use both terms, grade and level interchangeably). More importantly, every user can start with a secret key generated by one authority and then can upgrade the level of his secret key by sequentially contacting other authorities depending on their indices. The authority can provide some auxiliary information to the user so that the user can upgrade the level of his key.

We highlight some requirements of a GE scheme here that makes the primitive challenging to implement. The security requirement we emphasize is that a secret key with grade i can not be useful in decrypting a ciphertext with grade j when $j > i$; in our formalization we even consider the stronger requirement that, any authority

except the central authority can be corrupted as long as one authority in the chain of $1, 2, \ldots, i$ remains honest. We require that a ciphertext with grade i would remain secure in such conditions. This strong requirement is necessary in applications where we want to protect against an adversary who might directly corrupt the authority at level i while ignoring authorities corresponding to stage-1 to stage-$(i-1)$ in the hope of unlocking ciphertexts of level i.

We present an efficient scheme of graded encryption that has both secret key and ciphertext of constant size. We further extend it to a two-mode version that there are two types of partial keys that each authority can provide, and depending on that, there are two types of secret keys of each level. A Type-1 secret key is used to decrypt a ciphertext with the same level, but it can not be upgraded anymore; while a Type-2 secret key can be upgraded to higher level depending on the type of partial secret key, but it is not useful for decrypting ciphertexts. The two-mode GE is suitable for the application of online games like "who wants to be a millionaire", i.e., the Type-1 secret keys correspond to the choice of "walk away"; and a Type-2 partial key is used as long as the player chooses "continue" and advances levels (without receiving a reward).

6.1 Definition and Security Model

As elaborated above, we are considering a new cryptographic primitive we call graded encryption, which can be seen as an extension of the canonical identity based encryption [17, 85, 90]. In a graded encryption scheme, each ciphertext encrypted for an identity is associated with an integer j, called the grade of the ciphertext, according

to the importance of the content. Every secret key of an identity is also associated with an integer i, which we call the grade of the secret key. We require that a ciphertext encrypted under identity id with grade j can only be decrypted by secret key of id with the same grade. Furthermore, there is a mechanism that the user with identity id can upgrade his secret key with grade i to grade $(i+1)$ if the secret key with grade i was updated in same manner all the way from a key with grade 1.

We present the formal definition of a graded encryption scheme as the following five algorithms: Setup, KeyGen, Upgrade, Enc, and Dec.

- **Setup**(λ, n) This algorithm takes a security parameter λ and a maximum grade n as inputs and outputs the system parameters P, the master public keys $\{mpk_i\}_{i=1,\ldots,n}$ and the master secret keys denoted by $\{msk_i\}_{i=1,\ldots,n}$ where each (mpk_i, msk_i) correspond to a level i.

- **KeyGen**(id, i, msk_i, P) This algorithm takes user id and an integer i, the corresponding master secret key msk_i and the system parameter P as input, and outputs a partial secret key psk_i for grade i.

- Upgrade(sk_{i-1}, psk_i) This algorithm takes a secret key sk_{i-1} with grade $(i-1)$ and a partial secret key psk_i, and outputs the secret key sk_i for level i.

- **Enc**(m, mpk_i, id, P) This randomized algorithm takes the level i public key mpk_i, a message $m \in M$, where M is the message space, the user's identity id and the system parameter P as inputs. It outputs a ciphertext c_i with grade i.

- **Dec**(c_i, sk_i, P) This deterministic algorithm takes a ciphertext c_i with grade i encrypted under the identity id, the secret key sk_i with grade i for id and the system parameter P as inputs. It outputs message m.

We first define the correctness that using a secret key with grade i for identity id, one should be able to decrypt a grade i ciphertext encrypted under id, i.e.,: $\forall m \in M, \Pr[\mathbf{Dec}(\mathbf{Enc}(m, mpk_i, id, P), sk_i, P) = m] = 1$.

We now present the security model of a graded encryption scheme. From the exemplary applications in the introduction, we can see that the only way a user can decrypt a graded ciphertext with a grade i is if she executes a sequential update for her secret key from level 1 to level i. In our security model we allow the adversary to corrupt a set of authorities, and get partial secret key of an identity for any level. But we require that as long as there exists an $i_0 \leq i$ such that neither the adversary has the master secret key msk_{i_0} nor she has the partial secret key for the challenge identity in this level, she would not be able to have significant advantage decrypting a ciphertext with grade i. Thus, on top of the basic security requirement for any identity based encryption scheme, we have to capture this new security requirement.

We adapt the standard ID-IND-CPA security model for our purpose. There are two modified queries allowed for the adversary. *Corrupt queries* allow the adversary to get a master secret key corresponding to a level i of her choice; and *KeyGen queries* allow the adversary not only to get secret keys for other identities, but also allow her to learn some of the secret keys corresponding to certain grades for the challenge identity. Consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$,

- Setup. The challenger $\mathcal{C}$ runs the **Setup** algorithm and returns all the master public keys $\{mpk_i\}_{i=1,\ldots,n}$ and the maximum level n.
- The adversary is allowed to interleave the following queries.
  - Corrupt query. The adversary $\mathcal{A}$ ask a subset S of level master secret

keys, where $S \subset [1, n]$ is of her choice and with size q_1, and the challenger $\mathcal{C}$ returns $\{msk_i\}_{i \in S}$.

- KeyGen query. The adversary $\mathcal{A}$ asks a number of KeyGen queries for identities $id_1, \ldots, id_{q'}$ w.r.t. level $i_1, \ldots, i_{q'}$ of her choice, and the challenger returns $psk_{i_1}(id_1), \ldots, psk_{i_{q'}}(id_{q'})$, where $psk_i(id_j)$ is the KeyGen query on identity id_j for level i.

- Challenge. The adversary $\mathcal{A}$ sends an identity id^*, an integer i^*, and two messages m_0, m_1. The challenger flips a coin b and sends back $c_b = \mathbf{Enc}(m_b, mpk_{i^*}, id^*, P)$.

- The adversary continues asking Corrupt and KeyGen queries.

- Guess. The adversary outputs a guess b'.

We do allow the adversary to ask *KeyGen* queries for id^* w.r.t some levels, but we need to rule out attacks that trivialize the adversarial goal. Suppose S' is the subset of levels that $\mathcal{A}$ asks *KeyGen* queries for id^*. We require that $\{1, \ldots, i^*\} \setminus (\{1, \ldots, n\} \setminus S \cup S') \neq \emptyset$, i.e., there exists an $i_0 \leq i^*$, the adversary does not have msk_{i_0}, and also she does not have a partial secret key for id^* at level i_0.

Definition 4. If $b = b'$, the adversary wins. A graded encryption scheme is (t, q, q', ϵ)-fully ID-IND-CPA secure if all t-time adversaries making at most q corrupt queries and q' KeyGen queries have advantage at most ϵ in winning the above game.

In the game defined above, the adversary does not need to declare the target identity or the target grade at the beginning of the game, we call the above game a fully ID-IND-CPA game. Other possible variants can also be considered, i.e., if the adversary needs to declare the target identity at the beginning of the game, we call

it selective-ID IND-CPA game; if the adversary needs to declare the target grade at the beginning of the game, we call it selective-grade ID-IND-CPA game; if both of them need to be declared at the beginning, we call it selective IND-CPA game.

Remark that our security model explicitly states that there is no hierarchy among the master secret keys, having a master secret key with a higher grade does not grant the ability to decrypt a lower grade ciphertext.

6.2 Constructions of Graded Encryption

In this section, we set forth to construct efficient graded encryption schemes and briefly describe how we could deploy such schemes. First, we can have a simple generic construction of a graded encryption scheme from any IBE scheme by doing a message splitting. Briefly, in **Setup**, n independent master key pairs $\{(mpk_i, msk_i)\}_{i=1,\ldots,n}$ for each authority are generated by calling the Setup algorithm of the underlying IBE. To construct a ciphertext $c_i$with the grade i, the message is simply split into $m_1, \ldots, m_i$, and each part m_j is encrypted under the receiver's identity by using corresponding master public key mpk_j. A secret key with grade i consists of i partial secret key pk_j such that each of them is generated by calling the KeyGen algorithm of the underlying IBE. Furthermore, the decryption algorithm can be easily done by decrypting each piece of ciphertext using the corresponding key and combine the plaintexts to return the message.

This solution is easy and generic, however, it incurs large overhead (linear in the number of levels) of the secret key and the ciphertext for each level. A much more preferable solution would achieve a constant size for the secret key, ciphertext and

each master key pair.

Intuitively, if an IBE scheme has some kind of key homomorphism so that one can aggregate secret keys to reduce the key size; also, if the master public keys can be aggregated, then the ciphertext size could also be brought down to constant. Actually, most of the existing IBE schemes satisfy those requirements, thus we pick the Waters IBE [90] as an example to present the idea. We leave as an interesting open problem whether it is possible to have a black-box construction of a constant overhead graded encryption scheme from *any* IBE scheme. The detailed construction (we call S-I) based on Waters IBE is as follows:

- **Setup**(λ, n). The algorithm first chooses a random generator $g \in \mathcal{G}$ and a random element g_2 of $\mathcal{G}$. It also chooses a set of random numbers $a_1, \ldots, a_n \in Z_p$. For $i = 1, \ldots, n$, the algorithm computes $mpk_i = g^{\sum_{j=1}^{i} a_j}$, and $msk_i = g_2^{a_i}$. The algorithm also chooses a random value $u' \in G$ and a random l-length vector $U = (u_i)_{i=1,\ldots,\ell}$ where $u_i \in \mathcal{G}$ and ℓ is the length of the identities. It outputs the master key pairs $\{mpk_i, msk_i\}_{i=1,\ldots,n}$ and the system parameter $P = (g, g_2, u', U)$.

- **KeyGen**(id, i, msk_i, P). Suppose $V \subseteq [n]$ is the set of all indices j for which $id_j = 1$. The algorithm then chooses a random $r \in Z_p$, and computes psk_i as $(psk_i^1, psk_i^2) = (msk_i \cdot H(id)^r, g^r)$, where $H(id) = u' \prod_{i \in V} u_i$.

- Upgrade(sk_{i-1}, psk_i): If $i = 1$, $sk_1 = psk_1$; If $i > 1$, the algorithm computes $sk_i = (sk_i^1, sk_i^2)$ where $(sk_{i-1}^1 \cdot psk_i^1, sk_{i-1}^2 \cdot psk_i^2)$.

- **Enc**(id, m, mpk_i, P) To encrypt a message m under identity id for grade i, the

algorithm computes c_i as: (C_1, C_2, C_3) equal to:

$$(e(mpk_i, g_2)^s \cdot m, g^s, H(id)^s) = (e(g^{\sum_{j=1}^{i} a_j}, g_2)^s \cdot m, g^s, H(id)^s).$$

It returns (C_1, C_2, C_3) as the grade i ciphertext c_i.

- **Dec**(c_i, sk_i, P). Taking the level i ciphertext $c_i = (C_1, C_2, C_3)$ and level i secret key $sk_i = (sk_i^1, sk_i^2)$ as input, it computes $C_1 \cdot e(sk_i^2, C_3)/e(sk_i^1, C_2) = m$. It returns m as the plaintext.

Deployment: Based on our applications, our graded encryption scheme can be implemented as follows: a central authority initializes and sets up the system at the beginning, and distributes each level master keys to the individual authorities. Then, the central authority goes offline. The **KeyGen** algorithm is executed by each individual authority which only has a master secret corresponding to one level. When a user has a secret key with grade i, together with the partial secret key with grade $(i+1)$ generated by authority $(i+1)$, he can upgrade his key to grade $(i+1)$ locally. Observe that in the above construction, msk_i does not correspond to mpk, thus if the adversary does not update a secret key from grade 1, corrupting an intermediate authority does not give much advantage. Furthermore, the master secret key with grade i is independent of the master public keys with smaller grade, thus the online authorities can not decrypt the user's ciphertext as in the normal IBE setting unless all of them collude. This reveals an interesting fact that our graded encryption scheme provides some defense to the key escrow problem in IBE systems.

Theorem 6. S-I is fully ID-IND-CPA secure if Waters IBE [90] is ID-IND-CPA secure.

Proof. We here show how to reduce the security of S-I to that of Waters IBE.

The simulator $\mathcal{S}$ first makes a random guess for the index j of the sub-authority which the adversary will not corrupt, and $\mathcal{S}$ will abort if the guess is incorrect. For other indices, $\mathcal{S}$ selects a random a_i and generates the master key pair using (a_i, g^{a_i}) multiplying all g^{a_i} with the Waters IBE master public key, $\mathcal{S}$ gives the system mpk. Note that $\mathcal{S}$ knows msk for all $i \neq j$ thus can answer all Corrupt and KeyGen queries trivially, and KeyGen query for j can be answered by asking Waters IBE challenger secret key query.

In the challenge phase; the adversary submits two messages m_0, m_1 and an identity id^* with a grade i^*. if $j > i^*$, the simulator aborts. Otherwise, the simulator uses the Waters IBE challenge ciphertext (c_1, c_2, c_3) by asking m_0, m_1 on id^* to derive $c_i^{(b)} = (c_1.e(c_2, g_2)^{\sum_{k=1,k\neq j}^{i} a_k}, c_2, c_3)$ to the adversary as the challenge ciphertext where all a_k are integers generated at the beginning of the game for the master secret keys. Furthermore, it is easy to see that the probability of which the simulator aborts is bounded by $\frac{1}{n}$. □

6.3 Two-Mode Graded Encryption Schemes

In order to be applicable for the online quiz show like "who wants to be a millionaire", we need that our GE Upgrade algorithm supports two types of operations, i.e., decrypt now or continue. We revise the above construction that there are two **KeyGen** algorithms outputting two types of partial key. To be more specific, if a player wants to take the current rewards and leave, the authority prepares a partial secret key of Type-1, which together with the current key of the user, can be upgraded to a

Type-1 secret key to decrypt the reward ciphertext, but Type-1 secret key can not be upgraded anymore; if he decides to "continue", the authority prepares a partial secret key of Type-2, which can only be used to upgrade secret key to a upper level, and not useful decrypting current level ciphertext. Due to the nice property of our graded encryption construction, we can achieve such a scheme. We first show a modified construction of graded encryption scheme which has two **KeyGen** algorithms, we call the following construction S-II.

- **Setup**(λ, n). The algorithm first chooses a random generator $g \in \mathcal{G}$ and a random element g_2 of $\mathcal{G}$. It also chooses a set of random numbers $a_0, b_0 \ldots, a_n, b_n \in Z_p$. For $i = 0, \ldots, n$, the algorithm computes $mpk_i = g^{\sum_{j=0}^{i-1} b_j + a_i}$, and $msk_i^{(1)} = g_2^{a_i}, msk_i^{(2)} = g_2^{b_i}$. The algorithm also chooses a random value $u' \in G$ and a random l-length vector $U = (u_i)_{i=1,\ldots,\ell}$ where $u_i \in \mathcal{G}$ and ℓ is the length of the identities. It outputs the master key pairs $\{mpk_i, msk_i^{(1)}, msk_i^{(2)}\}_{i=0,\ldots,n}$ and the system parameter $P = (g, g_2, u', U)$.

- **KeyGen1**$(id, i, msk_i^{(1)}, P)$. Suppose $V \subseteq [n]$ is the set of all indices j for which $id_j = 1$. The algorithm then chooses a random $r \in Z_p$, and computes the type-1 partial secret key $psk_i^{(1)}$ as $(psk_{i,1}^{(1)}, psk_{i,2}^{(1)}) = (msk_i^{(1)} \cdot H(id)^r, g^r)$, where $H(id) = u' \prod_{i \in V} u_i$.

- **KeyGen2**$(id, i, msk_i^{(2)}, P)$. Suppose $V \subseteq [n]$ is the set of all indices j for which $id_j = 1$. The algorithm then chooses a random $r \in Z_p$, and computes the type-2 partial secret key $psk_i^{(2)}$ as $(psk_{i,1}^{(2)}, psk_{i,2}^{(2)}) = (msk_i^{(2)} \cdot H(id)^r, g^r)$, where $H(id) = u' \prod_{i \in V} u_i$.

- Upgrade$(sk_{i-1}, psk_i^{(b)})$: If $i = 0$, $sk_0 = psk_0$; If $i > 0$, for both $b = 1, 2$, the algorithm computes $sk_i = (sk_i^{(1)}, sk_i^{(2)})$ where $(sk_{i-1}^{(1)} \cdot psk_{i,1}^{(b)}, sk_{i-1}^{(2)} \cdot psk_{i,2}^{(b)})$.

- **Enc**(id, m, mpk_i, P) To encrypt a message m under identity id for grade i, the algorithm computes c_i as:

$$(C_1, C_2, C_3) = (e(mpk_i, g_2)^s \cdot m, g^s, H(id)^s) = (e(g^{\sum_{j=0}^{i-1} b_j + a_i}, g_2)^s \cdot m, g^s, H(id)^s).$$

  It returns (C_1, C_2, C_3) as the grade i ciphertext c_i.

- **Dec**(c_i, sk_i). Taking the level i ciphertext $c_i = (C_1, C_2, C_3)$ and the level i secret key $sk_i = (sk_{i,1}, sk_{i,2})$ as input, it computes $C_1 \cdot e(sk_{i,2}, C_3)/e(sk_{i,1}, C_2) = m$. It returns m as plaintext.

`Note` that the Type-1 secret key $sk_i^{(1)}$ is accumulated from $(i-1)$ Type-2 partial secret keys and with a Type-1 partial secret key as the last one. Besides, it corresponds to the exponent in the master public key, thus can be used for decrypting the ciphertext with level i. But we can also see that the Type-1 secret key cannot be upgraded anymore for next level as there is no way to eliminate the Type-1 partial secret key in the middle . While the Type-2 secret key is accumulated all by Type-2 partial secret keys, thus it can not be used for decryption unless upgrade at the end using some Type-1 partial secret key.

6.4 Applications of Graded Encryption Schemes

Our primitive is motivated by a bunch of interesting applications. In this section we will show in more detail how graded encryption schemes can be used to design

efficient or alternative solutions for those real world problems. We will focus on three applications: 1) graded rewarding system, 2) online quiz show like "who wants to be a millionaire", and 3) proving that a certain path was in a graph.

6.4.1 Graded rewarding system

In a graded reward system, there are multiple stages of tasks, and there are rewards corresponding to the task of each stage. Once the user completes a task, he should be given access to the corresponding reward. In many such systems, the tasks have hierarchy, i.e., a user has to finish task $1, \ldots, i$, respectively, in order to start task $(i + 1)$. The best example of such system is online video games: a user has to sequentially complete the tasks associated to different stages with certain difficulty to gain the experience points. He then uses these experience points to unlock some rewards, and proceed to the next level of challenge. Our graded encryption scheme will enable us to utilize a simple model that there is an individual authority who manages the tasks for each stage. Hence, corrupting each individual authority does not influence the security of rewards for other stages, nor propagates the damage outside this stage.

For the implementation, we will have the following system: there is a central authority and n authorities so that each of them manages each level i individually. The central authority initializes and sets up the system, and assigns associated keys into each authority correspondingly. Furthermore, the central authority also prepares a set of identities, and for each identity, produces the ciphertexts by encrypting all the rewards corresponding to each level. It then gives all identities together with the corresponding ciphertexts to the first level authority. Note that the central authority

will be offline in general. Only when the number of users exceeds the number of identities prepared at setup, it comes online to prepare another set of such ciphertexts, and becomes offline again.

A user starts his journey of conquering the challenges the game presents by registering the first level authority and getting a bunch of rewards locked in some ciphertexts. Once he finishes the task of stage i, the authority i verdicts the achievement and provides a partial "certificate" which can be used together with the "certificate" that the user currently holds, to unlock the corresponding reward. We assume here explicitly that there is an independent authentication mechanism that each authority can check whether the user does finish the task, and we will present a simple one in the next application.

A bit more formally, in a graded reward system, there are n stages of challenges, and for each user, there is a reward after finishing each challenge. There is a restriction for the application; the user can only start the challenge at stage i, if he already completed all tasks associated to previous levels. We use the following algorithms to represent a graded reward system: **Setup**, **Register**, **Proceed**, **Reward**. From functionality point of view, **Setup** algorithm computes the master keys of the authorities as $\{mpk_i, msk_i\}_{i=1,\ldots,n}$ for $i = 1, \ldots, n$, and prepares the identities id_j where $j = 1, \ldots, N$, for the identities id_j, computes the ciphertexts $c_i^{id_j}$ of rewards m_i correspoding to stage i; **Register** prepares the first level secret key of the user id_j, gives it to the user together with all ciphertexts $c_i^{id_j}$ containing the rewards m_i for $i = 1, \ldots, n$; **Proceed** algorithm allows a user to get a certificate psk_i after finishing a certain task and upgrade his secret key to next level; **Reward** algorithm provides a reward to the user depending on the task of the stage the user completes. In other words, after getting the secret key with level i, the user decrypts corresponding

ciphertext $c_i^{id_j}$ under his identity id_j, and gets the reward m_i.

From the security we discussed, given an i-level ciphertext the adversary cannot learn anything about the corresponding plaintext, even he corrupts all servers except one such that its index is smaller than i. So the player cannot reach the content of the ciphertext of level i if he fails on any stage from 0 to i even he is able to corrupt some of the intermediate authorities. Furthermore, each level ciphertext is composed of only constant number of group elements, and both the authorities and the players keep only constant size keys.

6.4.2 Online quiz shows

As the motivating example, we ask how to design an online quiz shows like the famous "who wants to be a millionaire". In such systems, the user has to answer questions with different level of difficulties. If the user provides a correct answer to a given question, he will be given the choice of "continue" or "leave with the current money". When the user chooses "leave", he can get the current reward and leave the game. But if he chooses "continue", instead of getting reward, he passes to the next stage to gain higher reward. However, he has to pass the next level challenge in order to get a chance to be able to choose again.

In order to support two types of operations explained above, we will use the extended construction. If the players decide to "continue", the authority prepares the partial secret key of type-2 by calling **KeyGen2**, which can only be used to upgrade secret key to a upper level, and not useful decrypting current level ciphertext. If they want to take the current rewards and leave, the authority prepares the partial secret key of type-1 by calling **KeyGen2**, which together with the current key of the user,

can be used to decrypt the reward ciphertext, furthermore, this key can not be upgraded. We also develop a simple authentication mechanism, in which each authority has a hash value of the answers, (we assume here the answers for each question are unique, the general case can be handled using a zero-knowledge proof system). Once the user figures out the answers, he provides the hash values to the authority to be able to make his choice of "continue" or "leave with the current rewards". Following the extended construction and the simple authentication mechanism, we can give a complete solution for the online quiz show problem:

A central authority initialize the system by running the **Setup** algorithm of the extended construction, and distributes to each individual authority the master secret keys $msk_i^{(1)}, msk_i^{(2)}$. Also as in the graded rewarding system, the central authority first prepares a set of user identities $\{id\}$ together with ciphertexts $\{c_i\}$ for the corresponding rewards by running $\mathbf{Enc}(id, m_i, mpk_i, P)$. Furthermore, the central authority prepares the questions for each level, and distributes the hash value of the answers h_i to each individual authority. The registration is the same as in the graded rewarding system. When the user starts the game and figures out the answer in level-i, he sends the hash of the answer h_i to the i-th authority together with his choice "leave" or "continue", then the authority will run **KeyGen1** and **KeyGen2** respectively, and returns the corresponding partial key to the user. The user runs the **Upgrade** algorithm to get his secret key for level-i. When the user finally decides to leave at level-j after providing true answer to the question, he uses his secret key with level j and runs the $\mathbf{Dec}(c_j, sk_j)$ to retrieve the level j accumulated rewards.

6.4.3 Proving a certain path was taken in a graph

An ordered authentication is needed in some applications like passport control, in which people should pass gate 1 to n one by one to get each check done. We should defend against the attack that a user jumps into the middle of the sequence say gate i by e.g., corrupting some of the gate controllers, and escapes the check from gate 1 to $i-1$. This is one of the applications of proxy re-signature, as introduced in [2], here we provide a simple alternative solution to this problem using our graded encryption. The key factor for this problem is to provide a proof that a user already pass the checkings of *all* previous gate controls. A simple observation is that the i-th gate controller can simply encrypts a random message r under the user identity, using the master public key with level $(i-1)$, and gives the encryption to the user. If the user's secret key is upgraded from level 0 to level $i-1$ already, he will be able to decrypt the encryption.

To be more specific, consider the following algorithms/protocols: **Setup** The central authority runs the **Setup** algorithm of the graded encryption scheme and distribute master public key mpk_{i-1} and master secret key msk_i to gate controller i. **Check** Whenever a user (say a traveller) arrives a checkpoint i, he first gives his identity to the gate controller i. The gate controller i encrypts a randomly chosen message r as $c_i = \mathbf{Enc}(id, r, mpk_{i-1}, P)$, sends it to the traveler. The traveler decrypts c_i as $m' = \mathbf{Dec}(c_i, sk_{i-1})$, and sends it back. If $m = m'$, then the traveler is considered to have passed all the checkpoints from 1 to $i-1$. The gate controller i then checks the criterial for gate i, if pass as well, he prepares a partial secret key $psk_i = \mathbf{KeyGen}(id, i, msk_i)$ and sends psk_i to the traveller. **Pass** After passing the checks from a gate controller i, the user upgrade his "certificate" using the partial

secret by running the **Upgrade** algorithm and moves on to the next gate.

The semantic security of graded encryption guarantees that the traveler needs all partial secret keys psk_j up to level i to construct the secret key sk_i. Therefore, holding a valid secret key of level i is enough to prove that the owner of the secret key passed all checkpoints from 1 to i. Besides, this mechanism is very efficient. The travelers upgrades his secret key whenever he gets a partial key. So he only keeps one secret key at a time. Moreover, each checkpoint keeps only two keys; one to check the validity of the secret key the traveler has, another to produce the partial secret key.

Chapter 7

Conclusions

In this thesis, we investigated a new functionality for privacy-preserving cryptosystems that enables a user to reveal only the number of sources he has contacted. Within this scope, we first formalized two new primitives: graded signatures, graded encryption; (i) graded signatures enables a user to combine a set of signatures on a message m originating from ℓ different signers into a signature object in a way that the final signature shows that ℓ distinct signers have signed on the message m, without revealing the identities of the signers, and (ii) graded encryption enables a sender to set a grade i to the ciphertext according to the importance of the plaintext in the encryption so that the receiver can only decrypt it if he has been approved by at least i authorities. We present efficient constructions for the primitives, i.e. the construction for GS has secret key and ciphertext size of a constant number of group elements, and the construction for GE achieves linear size signatures and verification time. We also promote the primitives with useful applications such as multi-stage games (e.g., "who wants to be a millionaire") played in a distributed fashion and

proving that a certain path was taken in a graph for graded encryption, and anonymous petition system and delegation of signing rights adhering to dynamic threshold policies for graded signatures.

We observed that distributing the signing keys of graded signature scheme associated to different grades among the signers in an efficient way, enables us to get a scheme with constant size signatures. Thus, we considered the signing keys as the secrets to be shared, and reviewed the problem of minimizing the share size of a multi-secret sharing scheme (MSSS) in the challenging setting when there are many secrets to be shared and there is no public "bulletin board." To circumvent the information-theoretic lower bound (Blundo [13]), we dealer with the computational setting. A simple generalization of computational secret sharing (Krawczyk [66]) to multi-secret sharing yields a scheme with share size/overhead scaling linearly in ℓ, the total number of secrets. To beat this linear scaling, we gave a construction of MSSS based on a related notion of encryption, dynamic threshold public key encryption (DTPKE)—that enables a sender to dynamically specify a threshold for each ciphertext. We first proposed a new construction of a dynamic threshold public key encryption scheme with improved efficiency characteristics. We then presented an efficient construction of MSSS with share size only logarithmic in the number of secrets (hence effectively optimal) based on the new DTPKE scheme

Bibliography

[1] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *CRYPTO*, pages 209–236, 2010.

[2] G. Ateniese and S. Hohenberger. Proxy re-signatures: new definitions, algorithms, and applications. In *ACM Conference on Computer and Communications Security*, pages 310–319, 2005.

[3] M. H. Au, S. S. M. Chow, W. Susilo, and P. P. Tsang. Short linkable ring signatures revisited. In *Public Key Infrastructure, Third European PKI Workshop: Theory and Practice, EuroPKI 2006, Turin, Italy, June 19-20, 2006, Proceedings*, pages 101–115, 2006.

[4] J. Baek and Y. Zheng. Identity-based threshold decryption. In *Public Key Cryptography*, pages 262–276, 2004.

[5] A. Beimel, A. Ben-Efraim, C. Padró, and I. Tyomkin. Multi-linear secret-sharing schemes. In *Theory of Cryptography-TCC 2014*, pages 394–418, 2014.

[6] M. Bellare and S. Goldwasser. Verifiable partial key escrow. In R. Graveman, P. A. Janson, C. Neumann, and L. Gong, editors, *CCS '97, Proceedings of the 4th*

ACM Conference on Computer and Communications Security, Zurich, Switzerland, April 1-4, 1997., pages 78–91. ACM, 1997.

[7] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS*, pages 390–399, 2006.

[8] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX 2014*, pages 781–796.

[9] J. Bethencourt, E. Shi, and D. Song. Signatures of reputation. In *Financial Cryptography*, pages 400–407, 2010.

[10] G. R. Blakley. Safeguarding cryptographic keys. *Managing Requirements Knowledge, International Workshop on*, page 313, 1979.

[11] O. Blazy, C. Chevalier, and D. Vergnaud. Non-Interactive Zero-Knowledge Proofs of Non-Membership. In *CT-RSA*, volume 9048, pages 145–164, 2015.

[12] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112, 1988.

[13] C. Blundo, A. D. Santis, G. D. Crescenzo, A. G. Gaggia, and U. Vaccaro. Multi-secret sharing schemes. In *CRYPTO '94, 1994*, pages 150–163, 1994.

[14] A. Boldyreva, A. Palacio, and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. *J. Cryptology*, 25(1):57–115, 2012.

[15] D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, pages 226–243, 2006.

[16] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[17] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[18] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.

[19] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 431–444, 2000.

[20] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *PKC 2014*, pages 501–519.

[21] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '02, pages 465–480, 2002.

[22] C. Cachin and S. Tessaro. Asynchronous verifiable information dispersal. In *DISC 2005*, pages 503–504, 2005.

[23] J. Camenisch, R. Chaabouni, and A. Shelat. Efficient protocols for set membership and range proofs. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 234–252, 2008.

[24] R. Canetti and S. Goldwasser. An efficient *Threshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106. Springer, 1999.

[25] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[26] R. Chaabouni, H. Lipmaa, and B. Zhang. A non-interactive range proof with constant communication. In *Financial Cryptography*, pages 179–199, 2012.

[27] Z. Chai, Z. Cao, and Y. Zhou. Efficient id-based broadcast threshold decryption in ad hoc network. In *Interdisciplinary and Multidisciplinary Research in Computer Science, IEEE CS Proceeding of the First International Multi-Symposium of Computer and Computational Sciences (IMSCCS|06), June 20-24, 2006, Zhejiang University, Hangzhou, China, Vol. 2*, pages 148–154, 2006.

[28] S. S. M. Chow, W. Susilo, and T. H. Yuen. Escrowed linkability of ring signatures and its applications. In *Progressin Cryptology - VIETCRYPT 2006, First*

International Conferenceon Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers, pages 175–192, 2006.

[29] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, pages 360–363, 2001.

[30] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *CCS '99, Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, November 1-4, 1999.*, pages 46–51, 1999.

[31] I. Damgård and M. Koprowski. Practical threshold rsa signatures without a trusted dealer. In *EUROCRYPT*, pages 152–165, 2001.

[32] V. Daza, J. Herranz, P. Morillo, and C. Ràfols. Cca2-secure threshold broadcast encryption with shorter ciphertexts. In *Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings*, pages 35–50, 2007.

[33] C. Delerablée and D. Pointcheval. Dynamic threshold public-key encryption. In *Advances in Cryptology - CRYPTO 2008*, pages 317–334, 2008.

[34] Y. Desmedt. Society and group oriented cryptography: A new concept. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, CRYPTO '87, pages 120–127, 1988.

[35] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 307–315, 1989.

[36] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[37] Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *TCC*, pages 188–209, 2005.

[38] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *EUROCRYPT*, pages 65–82, 2002.

[39] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *STOC '99*, pages 409–418, 1999.

[40] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[41] Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking and robust function sharing. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 499–508, 1996.

[42] G. Fuchsbauer and D. Pointcheval. Anonymous proxy signatures. In *SCN*, pages 201–217, 2008.

[43] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 16–30, 1997.

[44] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the tate pairing. In *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings*, pages 324–337, 2002.

[45] R. Gennaro, S. Halevi, H. Krawczyk, and T. Rabin. Threshold rsa for dynamic and ad-hoc groups. In *EUROCRYPT*, pages 88–107, 2008.

[46] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. Robust and efficient sharing of RSA functions. *J. Cryptology*, 13(2):273–300, 2000.

[47] C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[48] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC 87*, pages 218–229.

[49] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

[50] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17:281–308, April 1988.

[51] J. Groth. Non-interactive zero-knowledge arguments for voting. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 467–482, 2005.

[52] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 321–340, 2010.

[53] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

[54] J. Groth and A. Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

[55] J. Herranz, A. Ruiz, and G. Sáez. Sharing many secrets with computational provable security. *Inf. Process. Lett.*, 113(14-16):572–579, 2013.

[56] J. Herranz, A. Ruiz, and G. Sáez. New results and applications for multi-secret sharing schemes. *Des. Codes Cryptography*, 73(3):841–864, 2014.

[57] M. Hirt, C. Lucas, and U. Maurer. A dynamic tradeoff between active and passive corruptions in secure multi-party computation. In *CRYPTO 2013,*, pages 203–219, 2013.

[58] S. Hohenberger, V. Koppula, and B. Waters. Universal signature aggregators. In *EUROCRYPT 2015*, pages 3–34, 2015.

[59] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '02, pages 466–481, 2002.

[60] R. S.-N. Hossein Ghodosi, Joseph Pieprzyk. Dynamic threshold cryptosystems: A new scheme in group oriented cryptography. In *PRAGOCRYPT 1996*, pages 370–379.

[61] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Advances in Cryptology - CRYPTO 2006*, volume 4117, pages 483–500, 2006.

[62] W.-A. Jackson, K. M. Martin, and C. M. O'Keefe. Multisecret threshold schemes. In *CRYPTO 1993*, pages 126–135, 1993.

[63] D. Kaminsky, M. L. Patterson, and L. Sassaman. In *Proceedings of the 14th international conference on Financial Cryptography and Data Security*, FC'10, pages 289–303, 2010.

[64] E. D. Karnin, S. Member, J. W. Greene, S. Member, and M. E. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29:35–41, 1983.

[65] A. Kiayias, M. Osmanoglu, and Q. Tang. Graded encryption, or how to play "who wants to be A millionaire?" distributively. In *ISC 2014*, pages 377–387.

[66] H. Krawczyk. Secret sharing made short. In *CRYPTO '93*, pages 136–146, 1993.

[67] H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, pages 398–415, 2003.

[68] H. Lipmaa, N. Asokan, and V. Niemi. Secure vickrey auctions without threshold trust. In *Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda, March 11-14, 2002, Revised Papers*, pages 87–101, 2002.

[69] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, pages 325–335, 2004.

[70] L.Lamport. Constructing digital signatures from a one-way function. In *Technical Reprot SRI-CSL-98*, 1979.

[71] T. N. M. Itoh, A. Saito. Secret sharing scheme realizing general access structure. *IEEE Globecom*, page 99102, 1987.

[72] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *Proceedings of the 3rd ACM conference on Computer and communications security*, CCS '96, pages 48–57, 1996.

[73] B. Masucci. Sharing multiple secrets: Models, schemes and analysis. *Des. Codes Cryptography*, 39(1):89–111, 2006.

[74] U. M. Maurer and J. L. Massey. Cascade ciphers: The importance of being first. *J. Cryptology*, 6(1):55–61, 1993.

[75] R. C. Merkle. A certified digital signature. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 218–238, 1989.

[76] R. C. Merkle and M. E. Hellman. On the security of multiple encryption. *Commun. ACM*, 24(7):465–467, 1981.

[77] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 129–140, 1991.

[78] M. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36:335–348, 1989.

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

[80] R. L. Rivest, A. Shamir, and Y. Tauman. How to share a secret. *Communications of the ACM*, 22(22):612–613, 1979.

[81] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. ASIACRYPT '01, pages 552–565, 2001.

[82] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394, 1990.

[83] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC'14*, pages 475–484, 2014.

[84] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.

[85] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[86] V. Shoup. Practical threshold signatures. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'00, pages 207–220, 2000.

[87] C. Tartary, J. Pieprzyk, and H. Wang. Verifiable multi-secret sharing schemes for multiple threshold access structures. In *Information Security and Cryptology, Third SKLOIS Conference, Inscrypt 2007, Xining, China, August 31 - September 5, 2007, Revised Selected Papers*, pages 167–181, 2007.

[88] P. P. Tsang and V. K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *Information Security Practice and Experience, First International Conference, ISPEC 2005, Singapore, April 11-14, 2005, Proceedings*, pages 48–60, 2005.

[89] M. van Dijk, W.-A. Jackson, and K. M. Martin. A general decomposition construction for incomplete secret sharing schemes. *Des. Codes Cryptography*, (3):301–321, 1998.

[90] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[91] H. Wee. Threshold and revocation cryptosystems via extractable hash proofs. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 589–609, 2011.

[92] A. C. Yao. Protocols for secure computations (extended abstract). In *FOCS 82*, pages 160–164, 1982.

[93] K. Zhang. Threshold proxy signature schemes. In *ISW*, pages 282–290, 1997.