THRESHOLD CRYPTOGRAPHY WITH CHINESE REMAINDER THEOREM

A DISSERTATION SUBMITTED TO

THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Kamer Kaya

August, 2009

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

Asst. Prof. Dr. Ali Aydın Selçuk (Advisor)

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

Prof. Dr. Cevdet Aykanat

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

Assoc. Prof. Dr. Oya Ekin Karaşan

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

Dr. Cengiz Çelik

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

Assoc. Prof. Dr. Ali Doğanaksoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

THRESHOLD CRYPTOGRAPHY WITH CHINESE REMAINDER THEOREM

Kamer Kaya
Ph.D. in Computer Engineering
Supervisor: Asst. Prof. Dr. Ali Aydın Selçuk
August, 2009

Information security has become much more important since electronic communication is started to be used in our daily life. The content of the term information security varies according to the type and the requirements of the area. However, no matter which algorithms are used, security depends on the secrecy of a key which is supposed to be only known by the agents in the first place.

The requirement of the key being secret brings several problems. Storing a secret key on only one person, server or database reduces the security of the system to the security and credibility of that agent. Besides, not having a backup of the key introduces the problem of losing the key if a software/hardware failure occurs. On the other hand, if the key is held by more than one agent an adversary with a desire for the key has more flexibility of choosing the target. Hence the security is reduced to the security of the least secure or least credible of these agents.

Secret sharing schemes are introduced to solve the problems above. The main idea of these schemes is to share the secret among the agents such that only predefined coalitions can come together and reveal the secret, while no other coalition can obtain any information about the secret. Thus, the keys used in the areas requiring vital secrecy like large-scale finance applications and command-control mechanisms of nuclear systems, can be stored by using secret sharing schemes.

Threshold cryptography deals with a particular type of secret sharing schemes. In threshold cryptography related secret sharing schemes, if the size of a coalition exceeds a bound t, it can reveal the key. And, smaller coalitions can reveal no information about the key. Actually, the first secret sharing scheme in the literature is the threshold scheme of Shamir where he considered the secret as the constant

of a polynomial of degree $t-1$, and distributed the points on the polynomial to the group of users. Thus, a coalition of size t can recover the polynomial and reveal the key but a smaller coalition can not. This scheme is widely accepted by the researchers and used in several applications. Shamir's secret sharing scheme is not the only one in the literature. For example, almost concurrently, Blakley proposed another secret sharing scheme depending on planar geometry and Asmuth and Bloom proposed a scheme depending on the Chinese Remainder Theorem. Although these schemes satisfy the necessary and sufficient conditions for the security, they have not been considered for the applications requiring a secret sharing scheme.

Secret sharing schemes constituted a building block in several other applications other than the ones mentioned above. These applications simply contain a standard problem in the literature, the function sharing problem. In a function sharing scheme, each user has its own secret as an input to a function and the scheme computes the outcome of the function without revealing the secrets. In the literature, encryption or signature functions of the public key algorithms like RSA, ElGamal and Paillier can be given as an example to the functions shared by using a secret sharing scheme. Even new generation applications like electronic voting require a function sharing scheme.

As mentioned before, Shamir's secret sharing scheme has attracted much of the attention in the literature and other schemes are not considered much. However, as this thesis shows, secret sharing schemes depending on the Chinese Remainder Theorem can be practically used in these applications. Since each application has different needs, Shamir's secret sharing scheme is used in applications with several extensions. Basically, this thesis investigates how to adapt Chinese Remainder Theorem based secret sharing schemes to the applications in the literature. We first propose some modifications on the Asmuth-Bloom secret sharing scheme and then by using this modified scheme we designed provably secure function sharing schemes and security extensions.

Keywords: Threshold cryptography, secret sharing, function sharing, Asmuth-Bloom, Chinese Remainder Theorem, provable security.

ÖZET

ÇİN KALAN TEOREMİ'NE DAYALI EŞİK KRİPTOGRAFİSİ

Kamer Kaya
Bilgisayar Mühendisliği, Doktora
Tez Yöneticisi: Asst. Prof. Dr. Ali Aydın Selçuk
Ağustos 2009

Bilgi güvenliği, elektronik iletişimin hayatımızın her alanına girmesi ile birlikte giderek daha çok önemli hale gelmektedir. Bilgi güvenliği kavramının içeriği kullanıldığı uygulamanın çeşidine ve gereksinimlerine göre değişebilmektedir. Fakat kullanılan alan ya da uygulama ne olursa olsun, güvenlik için hangi algoritmalar kullanılırsa kullanılsın, güvenlik ilk önce gerekli kişilerin bilmesi gereken bir anahtarın gizli kalmasına dayanmaktadır.

Güvenliğin en önemli unsuru olan anahtarların gizli kalması ve kaybolmaması gereksinimleri değişik problemleri de beraberinde getirmektedir. Anahtarın sadece bir kişide, sunucuda ya da veritabanında saklanması, sistemin güvenliğini o kişinin güvenliğine ve güvenilirliğine indirgemektedir. Bunun yanında şifrenin başka bir kopyasının olmaması da yazılım/donanım arızaları gibi durumlarda anahtarın tamamen kaybedilmesi gibi sakıncalar içermektedir. Anahtarın birden fazla kişide bulunması durumunda ise anahtarı ele geçirmeye çalışan biri için artık bir değil birden fazla hedef vardır ve dolayısıyla, anahtarın güvenliği bu kişilerinin en az güvenliğe sahip olanının güvenliğine indirgenmektedir.

Anahtar paylaştırma yöntemleri ilk olarak yukarıda bahsedilen problemleri çözmek için önerilmiştir. Bu yöntemlerdeki ana fikir anahtarın belli bir grup içinde öyle paylaştırılmasıdır ki, sadece önceden belirlenen koalisyonlar bir araya geldiğinde anahtarı elde edebilmeli daha küçük koalisyonlar ise anahtar hakkında hiçbir bilgi elde edememelidir. Bu sayede, şirketlerin karar mekanizması uygulamaları, büyük ölçekli finans uygulamaları, nükleer sistemlerin komuta-kontrol uygulamaları gibi alanlarda gizli kalması gereken anahtarlar anahtar paylaştırma yöntemleri kullanılarak saklanabilir.

Eşik kriptografisi anahtar paylaştırma yöntemlerinin özel bir hali ile ilgilenir.

Eşik kriptografisine dayanan anahtar paylaştırma yöntemlerinde bir koalisyonun içindeki kişi sayısı, büyüklüğü, belli bir eşiği, kısaca t, geçiyorsa, o koalisyon anahtarı elde edebilir. Daha küçük koalisyonlar ise anahtar hakkında hiç bir bilgi elde edemezler. Literatürde ilk önerilen anahtar paylaştıma yöntemlerinden biri Shamir'in eşik kriptografisine dayanan yöntemidir. Shamir bu yöntemde anahtarı t-1 dereceli bir polinomun sabit terimi olarak düşünmüş ve polinomun geçtiği noktaları grup içinde dağıtmıştır. Bu sayede, gerekli olduğunda t büyüklüğündeki bir koalisyon, polinomu yaratarak anahtarı elde edebilir. Bu yöntem sonraları güvenlik üzerine araştırma yapan bilim insanları tarafından kabul görmüş ve değişik uygulamalarda kullanılmıştır. Bu yöntem ile yaklaşık aynı zamanlarda önerilen Blakley'in düzlem geometrisine dayalı anahtar paylaştırma yöntemi ve Asmuth ve Bloom'un önerdiği Çin Kalan Teoremi@ne dayalı yöntem güvenlik açısından gerekli ve yeterli şartları sağladıkları halde araştırmacılar tarafından rağbet görmemişlerdir.

Anahtar paylaştırma yöntemleri yukarıda bahsedilen uygulamalar dışında da değişik güvenlik uygulamaları için temel yapı parçacığı görevini görmüşlerdir. Bu uygulamalar, genelde fonksiyon paylaştırma yöntemi olarak bilinen, herhangi bir fonksiyonun çıktısının, herbiri gizli bir fonksiyon girdisine sahip bir grup tarafından, fonksiyon girdileri gizli kalmak şartı ile hesaplanması problemini içerir. Literatürde, anahtar paylaştırma yöntemleri temel alınarak paylaştırılan bu fonksiyonlara RSA, ElGamal ve Paillier gibi açık anahtar algoritmalarının imza yada şifreleme fonksiyonları örnek gösterilebilir. Elektronik seçim gibi yeni nesil uygulamalar fonksiyon paylaştırma yöntemlerini yoğun bir şekilde kullanmaktadır.

Daha önce de bahsedildiği gibi, Shamir'in anahtar paylaştırma yöntemi literatürde sıklıkla kullanılan bir yöntem olup diğer anahtar paylaştırma sistemleri pek rağbet görmemektedir. Fakat, bu tezin gösterdiği gibi Çin Kalan Teoremine dayalı anahtar paylaştırma yöntemleri de pratik olarak bu tür uygulamalarda kullanılabilir. Her uygulama değişik güvenlik gereksinimlerine sahip olduğu için, Shamir'in yöntemi değişik eklentiler tasarlanarak çeşitli uygulamalarda kullanılmıştır. Bu tez temel olarak farklı anahtar paylaştırma yöntemlerinin çeşitli uygulamalarda nasıl kullanabileceği üzerine yoğunlaşacaktır. Tezde Çin Kalan Teoremi'ne dayalı bir anahtar paylaştırma yöntemi olan Asmuth-Bloom yöntemi için bazı değişiklikler önerilecektir. Sonra da bu yeni yöntemler kullanılarak kanıtlanabilir güvenliğe sahip fonksiyon paylaştırma yöntemleri ve halihazırda

varolan uygulamalarda gereken değişik güvenlik eklentileri tasarlanacaktır.

Anahtar sözcükler: Eşik kriptografisi, anahtar paylaştırma, fonksiyon paylaştırma, Asmuth-Bloom, Çin Kalan Teoremi, kanıtlanabilir güvenlik.

Acknowledgement

Up to this moment, I always thought that this page will be the easiest to write. Now, I can see that on the contrary, it is the hardest because the people I will mention (and the ones I will forget to mention) in this page made the other parts easy for me.

Foremost, I would like to express my sincere gratitude to my supervisor and friend Dr. Ali Aydın Selçuk. Without his directions, I would have been lost in this area as a probable stranger doing random walks. He has always been more enthusiastic and motivated than me about our research. Fortunately, he already has a PhD degree so I am getting this one. I hope I will be an excellent supervisor like him.

I am thankful to my thesis committee Prof. Dr. Cevdet Aykanat and Prof. Dr. Oya Ekin Karaşan who spent their time to read my study reports and drafts. Their valuable comments significantly helped to improve the quality of this thesis. I also want to thank Dr. Cengiz Çelik and Assoc. Prof. Ali Doğanaksoy for reading the manuscript and their helpful comments.

During my PhD studies, I was a TUBİTAK scholar, and for their support I am also thankful to them.

I am also indebted to my f.r.i.e.n.d.s.: I want to thank, Funda, Özgün, Çiğdem, Serkan, Sengör, Engin, Murat, Alptuğ, Gökhan, and others that I forgot to mention. For their valuable friendship, support and understanding, I am grateful to σ(Ali Buğdaycı, Ata Türk, Oğuz Kurt, Özer Aydemir) for any permutation σ. Special thanks go to Duygu for being the sweetest and making me remember the quote "Carpe Diem".

And last but most of the my gratitude goes to my dearest family; my mother Beyhan, my father Ali, and my sisters Bahar and Pınar. Without their support, I would not be able to thank to anyone mentioned here because, this page, like all of the other pages, would not exist. To them, I dedicate this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In his seminal paper [67], Shamir quoted the following combinatorial problem:

Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed? What is the smallest number of keys to the locks each scientist must carry?

A simple combinatorial approach needs $(11, 6) = 462$ locks and $(10, 5) = 252$ for each scientist. Even if we could manufacture such a cabinet, what would we do if we had 111 or 1111 scientists? In cryptography, we have a similar problem called *secret sharing*. When a confidential information is cryptographically secured, a secret called the *key* is needed to access this information. Giving this key to only one person is not a good idea since he can lose the key and the information can be inaccessible. To solve this problem, the key can be shared among several people. Since the above combinatorial approach is not efficient and not practical, we need a secret sharing scheme to distribute the key among n people. Fortunately, threshold cryptography deals with the problem of sharing a highly sensitive secret among a group of n users so that only when a sufficient number t of them come together can the secret be reconstructed. Well-known secret sharing schemes (SSS) in the literature include Shamir [67] based on polynomial interpolation, Blakley [9] based on hyperplane geometry, and Asmuth-Bloom [2]

based on the Chinese Remainder Theorem.

A further requirement of a threshold cryptosystem can be that the subject function (e.g., a digital signature) should be computable without the involved parties disclosing their secret shares. This is known as the *function sharing problem*. A function sharing scheme (FSS) requires distributing the function's computation according to the underlying SSS such that each part of the computation can be carried out by a different user and then the partial results can be combined to yield the function's value without disclosing the individual secrets. Several protocols for function sharing [21, 22, 23, 24, 66, 68] have been proposed in the literature.

1.1 Secret Sharing Schemes

The problem of secret sharing and the first solutions were introduced independently by Shamir [67] and Blakley [9] in 1979. A (t, n)-secret sharing scheme is used to distribute a secret d among n people such that any coalition of size t or more can construct d but smaller coalitions cannot.

The first scheme for sharing a secret was proposed by Shamir [67] based on polynomial interpolation. To obtain a (t, n) secret sharing, a random polynomial $f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \ldots + a_0$ is generated over $\mathbb{Z}_p[x]$ where p is a prime number and $a_0 = d$ is the secret. The share of the ith party is $y_i = f(i)$, $1 \leq i \leq n$. If t or more parties come together, they can construct the polynomial by Lagrange interpolation and obtain the secret, but any smaller coalitions cannot.

Another interesting SSS is the scheme proposed by Blakley [9]. In a t dimensional space, a system of t non-parallel, non-degenerate hyperplanes intersect at a single point. In Blakley's scheme, a point in the t dimensional space (or, its first coordinate) is taken as the secret and each party is given a hyperplane passing through that point. When t users come together, they can uniquely identify the secret point, but smaller coalitions cannot.

A fundamentally different SSS is the scheme of Asmuth and Bloom [2], which shares a secret among the parties using modular arithmetic and reconstructs it by the Chinese Remainder Theorem (CRT).

1.1.1 Extensions on Threshold Secret Sharing Schemes

In the original secret sharing problem, we were trying to find a way that makes the secret available to sufficiently large coalitions. Hence, for security analysis, we allow an adversary to corrupt less than t users but no more. In this model, we assume that the adversary is honest but curious and he/she is not allowed to deviate from the protocol by impersonating a corrupted user. The secret sharing schemes above, described in their simplest forms, are secure under this adversary model. However, they do not have a false share detection mechanism if the adversary sends wrong shares in the reconstruction phase on behalf of a corrupted user. Furthermore, the adversary can also corrupt the dealer and in that case the users must check if their shares are consistent with the secret. This problem was proposed by Chor et al. in 1985 and with a *verifiable* secret sharing scheme (VSS) as the solution [17]. Formally, a VSS scheme provides mechanisms to users to verify their shares are consistent. Furthermore, in a VSS scheme, even if the dealer is corrupted, there is a well-defined secret that a valid coalition can reconstruct. After Chor et al., more efficient non-interactive verifiable secret sharing schemes were proposed by Feldman [27] and [59]. The security of the Feldman's scheme depends on the hardness of the discrete-logarithm problem whereas the Pedersen's scheme is information theoretically secure.

A further extension to verifiability is *public verifiability*. In a publicly verifiable secret sharing (PVSS) scheme, anyone, not only the users, can verify that the shares are consistent with each other. This property is included in Chor et al.'s scheme, which is the first VSS in the literature. However, both Feldman's and Pedersen's schemes do not satisfy public verifiability. In [69], Stadler introduced the PVSS enhancement and proposed two PVSS schemes where the security of the first one depends on the Decisional Diffie-Hellman (DDH) assumption, which we describe in Section 4.3. Other researchers also investigated the verifiability

extension such as [7, 28, 32, 64]

As described above, the secrecy of the key is guaranteed if the adversary is restricted to compromise less than t users throughout the entire life-time of the secret. If the secret key is valid for a long period of time, which is sufficient for t corruptions, a secret sharing scheme cannot protect the secret key. In [39], Herzberg et al. proposed a *proactive* secret sharing scheme, where the shares of the users are periodically renewed without changing the long-term secret. After this renewal operation, the previous shares become obsolete. Hence, in a proactive secret sharing scheme, an adversary needs to compromise at least t users in a time period, e.g., a day, a week or a month. Herzberg et al. also described the mechanisms to guarantee the integrity and the availability of the long-term secret. These mechanisms provide protocols to detect corrupted shares and recover them if necessary.

Another extension on SSS schemes is *threshold changeability*; with this extension, the threshold parameter t can be changed after the dealing phase. This problem was investigated by Martin et al. with the restriction that no secure channel exists between the dealer and the users [52]. Martin et al. solved two variations of this problem: when the dealer is available and when he is not. They proposed two constructions where the first one depends on the Shamir's SSS and the second one is geometrical. Later, Steinfeld et al. proposed lattice based approaches for the same problem: the first approach [70] was designed for CRT-based SSS schemes and the second approach [71] was designed for Shamir based SSS schemes.

1.1.2 Properties of Secret Sharing Schemes

A SSS is said to be *perfect* if coalitions with cardinality smaller than t cannot obtain *any* information on d; i.e., the cardinality of the set of secret candidates for d cannot be reduced by using $t-1$ or fewer shares. According to this definition, the secret sharing schemes described above, Shamir, Blakley and Asmuth-Bloom SSSs, are perfect SSSs. A stronger definition of perfectness is as follows: in a

perfect SSS, for an adversary with $t-1$ compromised shares, each secret candidate has the same probability for being the secret. Shamir's and Blakley's SSSs satisfy this kind of perfectness, however, as described in Section 2.1 and as Quisquater et al. shows [63], the original Asmuth-Bloom SSS does not satisfy it. Asmuth and Bloom only showed that when an adversary has $t-1$ shares, the entropy of the secret does not decrease too much [2].

A SSS is said to be *ideal* if all secrets have the same length as the secret. Shamir's and Blakley's SSSs are ideal secret sharing schemes but Asmuth-Bloom scheme can only be said almost ideal. Quisquater et al. showed that when the moduli in Asmuth-Bloom SSS are consecutive primes, the scheme is asymptotically ideal [63].

1.2 Function Sharing Schemes

In a (t, n) function sharing scheme, a key-dependent function is distributed among n people such that any coalition of size t or more can evaluate the function but smaller coalitions cannot. When a coalition S is gathered to evaluate the function, the ith user in S computes his own partial result by using his share y_i and sends it to the combiner to evaluate the overall result. The combiner must be honest while combining the partial results but can be curious and try to find the secret shares. Hence a function sharing scheme cannot reveal the users' secret shares to the combiner.

FSSs are typically used to distribute the private key operations in a public key cryptosystem (i.e., the decryption and signature operations) among several parties. Sharing a private key operation in a threshold fashion requires first choosing a suitable SSS to share the private key. Then the subject function must be arranged according to this SSS such that combining the partial results from any t parties will yield the operation's result correctly. This is usually a challenging task and requires some ingenious techniques.

In its simplest form, function sharing problem was investigated by several

researchers such as Desmedt [20] and Goldreich et al. [37]. However the proposed solutions in these works are impractical and interactive. After these works, the function sharing problem was formally introduced by Desmedt and Frankel in 1989 [22]. They also proposed non-interactive and practical threshold function sharing schemes for ElGamal encryption scheme. The solutions they proposed were based on Shamir's and Blakley's SSSs.

After Desmedt and Frankel's work, the function sharing problem for RSA public-key cryptosystem was investigated by several researchers where Shamir's SSS was the main tool. The additive nature of the Lagrange interpolation used in the combiner phase of Shamir's scheme makes it a suitable choice for function sharing, but it also provides several challenges, especially for RSA scheme. One of the most significant challenges is the computation of inverses in $\mathbb{Z}_{\phi(N)}$ for sharing the RSA function where $\phi(N)$ should not be known by the users. The first solution to this problem was proposed by Desmedt and Frankel [21], which solved the problem by making the dealer compute all potentially needed inverses at the setup time and distribute them to users mixed with the shares. A more elegant solution was found a few years later by De Santis et al. [66]. They carried the arithmetic into a cyclotomic extension of $\mathbb{Z}$, which enabled computing the inverses without knowing $\phi(N)$. Finally, a very practical and ingenious solution was given by Shoup [68] where he removed the need of taking inverses in Lagrange interpolation altogether.

Shoup's practical RSA scheme inspired similar works on different cryptosystems. Fouque et al. [29] proposed a similar threshold solution for the Paillier cryptosystem and used it in e-voting and lottery schemes. Later, Lysyanskaya et al. [50] improved this work and obtained a threshold Paillier encryption scheme secure under the adaptive security model.

Although using Shamir's SSS for sharing the ElGamal signature and decryption functions has its own unique problems, the computation of inverses in the exponent is relatively easier than that in RSA since all of the operations are done in mod p where p is a public prime hence $\phi(p) = p - 1$ is also public. As mentioned above, Desmedt and Frankel solved the function sharing problem in

1989 for ElGamal decryption function. However, an ElGamal based threshold signature was not proposed until 1996. In [34], Gennaro et al. proposed the first efficient threshold scheme for the Digital Signature Standard.

As a summary, several solutions for sharing the RSA, ElGamal and Paillier private key operations have been proposed in the literature [21, 22, 23, 24, 29, 34, 50, 66, 68]. The proposed solutions in these works are usually based on Shamir's SSS. To the best of our knowledge, before our work, no secure FSS has been proposed for any of the cryptosystems mentioned above.

1.2.1 Extensions on Function Sharing Schemes

Since FSSs are based on SSSs, the extensions on secret sharing schemes can also be described for function sharing schemes. For example, the robustness extension is similar to the verifiability extension described in Section 1.1.1. We say that a FSS is *robust* if it can withstand participation of corrupt users in the function evaluation phase. The general approach to achieve robustness in function sharing schemes is sending more information along with the partial result. In that approach, each user in the coalition sends a proof of correctness of his partial result. In robust FSS schemes, a valid proof cannot be generated by a user unless he has the correct share and the partial result is correct. In 1996, Gennaro et al. proposed robust threshold RSA schemes [33] and a robust DSS signature scheme [34] (improved and extended versions of these works by Gennaro et al. can be found in [35, 36]).

Similar to verifiability, the proactivity extension described above also has a counterpart in FSSs. A proactive approach can be used for FSSs by designing protocols for periodic refreshment and integrity protection of local shares. With this approach, the adversary will have only a short period of time to corrupt t users and obtain their shares. In 1997, Herzberg et al. introduced the proactivity problem and proposed proactive public key and signature schemes [38]. Their approach can be used for several discrete log cryptosystems such as DSS and Schnorr signatures, ElGamal-like signatures and encryption.

1.3 Contributions and Outline

Nearly all existing solutions for function sharing are based on the Shamir SSS [67]. On the other hand, using CRT-based solutions for secret and function sharing schemes has not been well-investigated. This thesis investigates how various primitives and schemes in threshold cryptography can be securely realized by using a CRT-based secret sharing scheme, i.e., Asmuth-Bloom SSS.

In Chapter 2, the original Asmuth-Bloom SSS is presented and its modified versions which are more suitable for function sharing schemes are proposed. We also propose a scheme for the non-threshold case that uses Asmuth-Bloom SSS to share a secret among n people such that only predefined coalitions, which are members of an access structure can reconstruct the secret.

Chapter 3 (based on [44], [49] and [43]) presents the threshold RSA scheme based on the CRT. We also adapt our ideas used in the threshold RSA scheme to propose threshold ElGamal and Paillier schemes. All of these schemes are provably secure against an adversary with $t-1$ shares where t is the threshold. At last we give a description of CRT-based threshold Naccache-Stern cryptosystem.

The Digital Signature Standard (DSS) is the current U.S. standard for digital signatures. In Chapter 4 (based on [48]) we investigate that how the DSS signature function can be shared by using the Asmuth-Bloom SSS. We propose a threshold DSS scheme and prove that the scheme is secure against an adversary with $t-1$ shares.

Secret and function sharing schemes can be enhanced by using various extensions: As described in Section 1.1.1, we call a SSS *verifiable* if each user can verify the correctness of his share in the dealing phase and no user can lie about his share in the reconstruction phase. Another extension, *proactivity*, makes a SSS capable of renewing the shares of the users without changing the long term secret such that any shares obtained by a corrupted party before the renewal phase become obsolete. For function sharing schemes, we say that a FSS is *robust* if it can withstand participation of corrupt users in the function evaluation phase. In

Chapter 5 (based on [45], [46], and [47]), we propose CRT-based verifiable and proactive secret sharing schemes, and robust function sharing schemes.

We conclude the thesis with Chapter 6.

Chapter 2

Asmuth-Bloom Secret Sharing Scheme

The Asmuth-Bloom secret sharing scheme is proposed by Asmuth and Bloom in 1983. Let n be the number of total users and t be the threshold, i.e., the size of the smallest coalition that can reconstruct the secret. The original Asmuth-Bloom SSS is close to perfect, i.e., for an adversary with $t-1$ shares, the secret can be any candidate from the secret domain. However, the probabilities of the secret being equal to two different candidates are different where the difference is non-negligible. To make the function sharing schemes in this thesis provably secure, this difference must be negligible to make these two different candidates indistinguishable from adversary's point of view. Hence, the original scheme needs to be modified to make it suitable for function sharing. In this chapter, first the original scheme and then the required modifications are presented.

The CRT based schemes such as Asmuth-Bloom can also be used for generic secret sharing. In such schemes, the valid coalitions which can reconstruct the secret are defined by an access structure. Unlike the threshold case, any subset of n users can be an element of the access structure and its cardinality need not to be bigger than a threshold. We will conclude this chapter by describing a recent scheme proposed by Bozkurt [13] which is based on the Asmuth-Bloom SSS and

can be used for any access structure.

2.1 The Original Scheme

Dealing and reconstructing the secret in the Asmuth-Bloom SSS are described in Fig. 2.1.

- *Dealer Phase:* To share a secret d among a group of n users, the dealer does the following:
  - A set of pairwise relatively prime integers $m_0 < m_1 < m_2 < \ldots < m_n$, where $m_0 > d$ is a prime, are chosen such that
  $$\prod_{i=1}^{t} m_i > m_0 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{2.1}$$
  - Let M denote $\prod_{i=1}^{t} m_i$. The dealer computes
  $$y = d + Am_0$$
  where A is a positive integer generated randomly subject to the condition that $0 \leq y < M$.
  - The share of the ith user, $1 \leq i \leq n$, is
  $$y_i = y \bmod m_i.$$
- *Combiner Phase:* Assume S is a coalition of t users gathered to construct the secret. Let M_S denote $\prod_{i \in S} m_i$.
  - Given the system
  $$y \equiv y_i \pmod{m_i}$$
  for $i \in S$, find y in $\mathbb{Z}_{M_S}$ using the Chinese Remainder Theorem.
  - Compute the secret as
  $$d = y \bmod m_0.$$

Figure 2.1: The Asmuth-Bloom secret sharing scheme.

According to the Chinese Remainder Theorem, y can be determined uniquely

in $\mathbb{Z}_{M_S}$. Since $y < M \leq M_S$, the solution is also unique in $\mathbb{Z}_M$.

The Asmuth-Bloom SSS is close to perfect in the sense that $t-1$ or fewer shares do not narrow down the key space: Assume a coalition S' of size $t-1$ has gathered and let y' be the unique solution for y in $Z_{M_{S'}}$. According to (2.1), $M/M_{S'} > m_0$, hence $y' + jM_{S'}$ is smaller than M for $j < m_0$. Since $\gcd(m_0, M_{S'}) = 1$, all $(y' + jM_{S'}) \bmod m_0$ are distinct for $0 \leq j < m_0$, and there are m_0 of them. That is, d can be any integer from $\mathbb{Z}_{m_0}$. However, this scheme is not exactly perfect since when $t-1$ shares are known, the key candidates are not equally likely. We refer the reader to a recent work by Quisquater et al. [63] for a detailed security analysis of Asmuth-Bloom and some other Chinese Remainder Based SSSs.

2.2 The Modified Asmuth-Bloom Scheme

Several changes were needed on the basic Asmuth-Bloom scheme to make it more suitable for function sharing. In this section we describe these modifications:

In the original Asmuth-Bloom SSS, the authors proposed an iterative process to solve the system $y \equiv y_i \pmod{m_i}$. Instead, we use a non-iterative and direct solution as described in [25], which turns out to be more suitable for function sharing in the sense that it does not require interaction between parties and has an additive structure which is convenient for exponentiations. Suppose S is a coalition of t users gathered to construct the secret d.

1. Let $M_{S\setminus\{i\}}$ denote $\prod_{j\in S, j\neq i} m_j$ and $M'_{S,i}$ be the multiplicative inverse of $M_{S\setminus\{i\}}$ in $\mathbb{Z}_{m_i}$, i.e.,
$$M_{S\setminus\{i\}} M'_{S,i} \equiv 1 \pmod{m_i}.$$
First, the ith user computes
$$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S.$$
2. y is computed as
$$y = \sum_{i\in S} u_i \bmod M_S.$$

3. The secret d is computed as

$$d = y \bmod m_0.$$

We note that, in the Asmuth-Bloom SSS, m_0 need not be a prime, and the scheme works correctly for a composite m_0 as long as m_0 is relatively prime to m_i, $1 \leq i \leq n$. Also note that m_0 need not be known during the secret construction process until the 3rd step above.

We also modified (2.1) as

$$\prod_{i=1}^{t} m_i > {m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{2.2}$$

in order to use it securely in the proposed FSSs. Note that equation (2.2) guarantees that d can be any integer from $\mathbb{Z}_{m_0}$ when $t-1$ or fewer shares are revealed.

Theorem 2.2.1. *For a passive adversary with $t-1$ shares in the modified Asmuth-Bloom scheme, every candidate for the secret is equally likely, i.e., the probabilities $\Pr(d = d')$ and $\Pr(d = d'')$ are approximately equal for all $d', d'' \in \mathbb{Z}_{m_0}$.*

Proof. Suppose the adversary corrupts $t-1$ users and just observes the inputs and outputs of the corrupted users without controlling their actions, i.e., the adversary is honest in user actions but curious about the secret. Let S' be the adversarial coalition of size $t-1$, and let y' be the unique solution for y in $Z_{M_{S'}}$. According to (2.1), $M/M_{S'} > m_0$, hence $y' + jM_{S'}$ is smaller than M for $j < m_0$. Since $\gcd(m_0, M_{S'}) = 1$, all $(y' + jM_{S'}) \bmod m_0$ are distinct for $0 \leq j < m_0$, and there are m_0 of them. That is, d can be any integer from $\mathbb{Z}_{m_0}$. For each value of d, there are either $\lfloor M/(M_{S'}m_0) \rfloor$ or $\lfloor M/(M_{S'}m_0) \rfloor + 1$ possible values of y consistent with d, depending on the value of d. Hence, for two different integers in $\mathbb{Z}_{m_0}$, the probabilities of d equals these integers are almost equal. Note that $M/(M_{S'}m_0) > m_0$ and given that $m_0 \gg 1$, all d values are approximately equally likely. □

Note that the new equation (2.2) makes the scheme asymptotically perfect but it also increases the share sizes. Hence the modified scheme is not ideal. But the size of a share in the new scheme is two times larger than the one in the original scheme hence the modified scheme is practical and it can be used in various applications. For all of the schemes, m_i, $1 \leq i \leq n$, are known by all users, but m_0 is kept secret by the dealer for some function sharing schemes such as threshold RSA and Paillier schemes described in Chapter 3.

2.3 Asmuth-Bloom SSS for General Access Structures

The secret sharing problem also arises for the general case: the secret is shared among n participants such that only a specified set of authorized coalitions can reconstruct the secret [8, 42]. Unlike the threshold case, the size of an authorized coalition is not important and can be equal to any integer from 1 to n. Let $\mathcal{P} = \{1, \ldots, n\}$ be the set of participants. The set of authorized coalitions $\Gamma \subset 2^{\mathcal{P}}$ is called the *access structure*. Note that Γ is monotonically increasing, i.e.,

$$\mathcal{A} \in \Gamma \implies \mathcal{B} \in \Gamma, \ \forall \mathcal{B} \supset \mathcal{A}.$$

We denote the *basis* of Γ, i.e., the set of minimal elements in Γ, with Γ_0. Hence

$$\mathcal{A} \in \Gamma_0 \implies \nexists \mathcal{B} \in \Gamma_0, \text{ such that } \mathcal{B} \supset \mathcal{A}.$$

The set of unauthorized coalitions $\Delta \subset 2^{\mathcal{P}}$ is called the *adversary structure*. Note that Δ is monotonically decreasing, i.e.,

$$\mathcal{A} \in \Delta \implies \mathcal{B} \in \Delta, \ \forall \mathcal{B} \subset \mathcal{A}.$$

We denote the set of maximal elements in Δ, with Δ_1. Hence

$$\mathcal{A} \in \Delta_1 \implies \nexists \mathcal{B} \in \Delta_1, \text{ such that } \mathcal{B} \subset \mathcal{A}.$$

It is obvious that $\Delta \cap \Gamma = \emptyset$. Note that the threshold case with threshold t can be represented as

$$\Gamma_0 = \{\mathcal{A} \in 2^{\mathcal{P}} : \ |\mathcal{A}| = t\},$$
$$\Delta_1 = \{\mathcal{A} \in 2^{\mathcal{P}} : \ |\mathcal{A}| = t - 1\}.$$

2.3.1 Multipartite Access Structures

Let $\mathcal{P}$, the set of users, be partitioned into r disjoint sets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_r$. Each set $\mathcal{X}_i$ has n_i users and $\sum_{i=1}^{r} n_i = n$. An access structure is *multipartite* when all users in a given class play the same role. Let σ be a random permutation of numbers 1 to n. Formally, we call an access structure Γ is r-partite if $\sigma(\Gamma) = \Gamma$ for any permutation σ such that $\sigma(\mathcal{X}_i) = \mathcal{X}_i$ for $i = \{1, 2, \ldots, r\}$.

Note that every access structure is multipartite since we can take $r = n$ and $\mathcal{X}_i = \{i\}$ for $i = \{1, 2, \ldots, n\}$. But we are usually interested with the smallest possible r value to characterize the access structure. A simple algorithm that checks

$$\Gamma \stackrel{?}{=} swap(\Gamma, i, j)$$

for each user pair i, j is sufficient to find the smallest r where $swap(\Gamma, i, j)$ swaps the user i and j in Γ and returns the resulting access structure. If the equality holds, it is obvious that users i and j should be in the same partition.

Let $\omega : 2^{\mathcal{P}} \rightarrow (\mathbb{Z}_{n_1+1} \times \mathbb{Z}_{n_2+1} \times \ldots \times \mathbb{Z}_{n_r+1})$ be a function such that

$$\omega(\mathcal{A}) = (|\mathcal{A} \cap \mathcal{X}_1|, |\mathcal{A} \cap \mathcal{X}_2|, \ldots, |\mathcal{A} \cap \mathcal{X}_r|).$$

Since the users in the same class play the same role, an access structure Γ can be uniquely represented as a set of r-ary vectors

$$\Omega(\Gamma) = \{\omega(\mathcal{A}) : \mathcal{A} \in \Gamma\}.$$

2.3.2 Asmuth-Bloom SSS for Multipartite Access Structures

The Asmuth-Bloom secret sharing scheme can be also used for general access structures. Here we will describe a scheme by [13] which uses Asmuth-Bloom SSS for secret sharing in multipartite access structures. Let us first recall that the modified Asmuth-Bloom SSS is perfect. For every part $\mathcal{X}_i$, we choose the moduli $m_0 < m_{1,i} < m_{2,i} < \ldots, < m_{n_i,i}$ as consecutive primes such that they satisfy

$$\prod_{j=1}^{\lfloor n_i/2 \rfloor} m_{j,i} > m_0 \prod_{j=1}^{\lfloor n_i/2 \rfloor - 1} m_{n_i-j+1,i}. \tag{2.3}$$

Note that (2.3) is the inequality used for a $(\lfloor n_i/2 \rfloor, n_i)$ threshold secret sharing. Also 2.3 implies

$$\prod_{j=1}^{t} m_{j,i} > m_0 \prod_{j=1}^{t} m_{n_i-j+1,i}$$

for every value of t. Hence, the moduli can be used for a (t, n_i)-secret sharing scheme for $1 \leq t \leq n_i$.

In the SSS described below, we will use $\Omega(\Gamma_0)$ instead of $\Omega(\Gamma)$ and show that any coalition in Γ can reconstruct the secret. We say that a r-ary vector $(K_1, K_2, \ldots, K_r)$ dominates the vector $(k_1, k_2, \ldots, k_r)$ if $K_i \geq k_i$ for $1 \leq i \leq r$. The SSS is given in Fig. 2.2.

In Fig. 2.2, each user in $\mathcal{X}_i$ has a share for each vector in $\Omega(\Gamma_0)$ if the ith element of the vector is nonzero. $S \in \Gamma$ if and only if there exists a vector in $\Omega(\Gamma_0)$ dominated by $(K_1, K_2, \ldots, K_r)$ where $K_i = |S \cap \mathcal{X}_i|$. The users in $S \cap \mathcal{X}_i$ can construct d_i since d_i is shared with a (k_i, n_i)-secret sharing scheme and $K_i \geq k_i$.

Let S' be an adversarial coalition. Let $K'_i = |S' \cap \mathcal{X}_i|$. Since $S' \notin \Gamma$, $(K'_1, K'_2, \ldots, K'_r)$ cannot dominate a vector in $\Omega(\Gamma_0)$. Hence for each vector in $\Omega(\Gamma_0)$, there exists at least one i such that $k_i > K'_i$. Since Asmuth-Bloom SSS is perfect, S' cannot obtain any information on at least one d_i, for each vector. Hence the scheme given in Fig. 2.2 is also perfect.

Dealer Phase: Let Γ be an r-partite access structure. To share a secret d according to Γ, the dealer does the following:

- For each r-ary vector $(k_1, k_2, \ldots, k_r) \in \Omega(\Gamma_0)$ the dealer
  - chooses random $d_i \in \mathbb{Z}_{m_0}$s for $1 \leq i \leq r$ where $\sum_{i=1}^{r} d_i \equiv d \bmod m_0$.
  - shares d_i among the users in $\mathcal{X}_i$ by using a (k_i, n_i) Asmuth-Bloom secret sharing scheme.

Combiner Phase: Assume S is the coalition gathered to construct the secret. Let $K_i = |S \cap \mathcal{X}_i|$ and $(k_1, k_2, \ldots, k_r) \in \Omega(\Gamma_0)$ be a vector dominated by $(K_1, K_2, \ldots, K_r)$.

- For each $1 \leq i \leq r$, the users from $\mathcal{X}_i$ in S can construct the corresponding d_i for the vector $(k_1, k_2, \ldots, k_r)$ since $K_i \geq k_i$.
- The secret d is the constructed by

$$d = \sum_{i=1}^{r} d_i \bmod m_0.$$

Figure 2.2: Using Asmuth-Bloom SSS for general access structures.

Chapter 3

Sharing RSA and Similar Functions with CRT

In this chapter, we show how sharing of cryptographic functions can be securely achieved using the Asmuth-Bloom secret sharing scheme. We give four novel FSSs, one for the RSA [65], one for the ElGamal decryption [26], one for the Paillier decryption [57], and the other for the Nacceche-Stern decryption [54] functions. These public key cryptosystems have several interesting properties useful in various applications [1, 4, 29, 51, 56]. The proposed schemes are provably secure and to the best of our knowledge they are the first realizations of secure function sharing based on the Asmuth-Bloom SSS.

3.1 CRT-based Threshold RSA Scheme

RSA [65] is the first and the most commonly used public key cryptosystem today. Here we show how the RSA signature and decryption functions can be shared by using the Asmuth-Bloom SSS. Below, we limit our discussion to the RSA signature function since these two functions are identical and the same technique can be applied for sharing the decryption function as well. The description of the RSA signature scheme is given in Fig. 3.1.

- *Setup*: Let $N = pq$ be the product of two large prime numbers. Choose a random $e \in \mathbb{Z}^*_{\phi(N)}$ and find its inverse d, i.e., $ed \equiv 1 \pmod{\phi(N)}$. The public and private keys are (N, e) and d, respectively.
- *Signing*: Given a hashed message $w \in \mathbb{Z}_N$, the signature s is computed as

$$s = w^d \bmod N.$$

- *Verification*: Given a signature $s \in \mathbb{Z}_N$, the verification is done by checking

$$w \stackrel{?}{=} s^e \bmod N.$$

Figure 3.1: The RSA signature scheme.

Threshold RSA Signature Scheme: The following is a procedure that shares the RSA signature function among n users with the Asmuth-Bloom SSS such that when t users come together they can compute the signature:

- *Setup*: In the RSA setup phase, choose the RSA primes $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are also large random primes. $N = pq$ is computed and the public key e and private key d are chosen from $\mathbb{Z}^*_{\phi(N)}$ where $ed \equiv 1 \pmod{\phi(N)}$. Use Asmuth-Bloom SSS for sharing d with $m_0 = \phi(N) = 4p'q'$.
- *Signing*: Let w be the hashed message to be signed and suppose the range of the hash function is $\mathbb{Z}^*_N$. Assume a coalition S of size t wants to obtain the signature $s = w^d \bmod N$.
  - *Generating partial results*: Each user $i \in S$ computes

$$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S,$$
$$s_i = w^{u_i} \bmod N.$$

  - *Combining partial results*: The incomplete signature $\overline{s}$ is obtained by combining the s_i values

$$\overline{s} = \prod_{i \in S} s_i \bmod N. \tag{3.1}$$

– *Correction*: Let $\kappa = w^{-M_S} \bmod N$ be the *corrector*. The incomplete signature can be corrected by trying

$$(\overline{s}\kappa^j)^e = \overline{s}^e(\kappa^e)^j \stackrel{?}{\equiv} w \pmod{N} \tag{3.2}$$

for $0 \leq j < t$. Then the signature s is computed by

$$s = \overline{s}\kappa^\delta \bmod N$$

where δ denotes the value of j that satisfies (3.2).

- *Verification* is the same as the standard RSA verification.

We call the signature $\overline{s}$ generated in (3.1) *incomplete* since we need to obtain $y = \sum_{i \in S} u_i \bmod M_S$ as the exponent of w. Once this is achieved, we have $w^y \equiv w^d \pmod{N}$ as $y = d + Am_0$ for some A where $m_0 = \phi(N)$.

Note that the equality in (3.2) must hold for some $j \leq t-1$ since the u_i values were already reduced modulo M_S. So, combining t of them in (3.1) will give $d + am_0 + \delta M_S$ in the exponent for some $\delta \leq t-1$. Thus in (3.1), we obtained

$$\overline{s} = w^{d+\delta M_S} \bmod N = sw^{\delta M_S} \bmod N = s\kappa^{-\delta} \bmod N$$

and for $j = \delta$, equation (3.2) will hold. Also since $\phi(N) \gg t$, with overwhelming probability, there will be a unique value of $s = \overline{s}\kappa^j$ which satisfies (3.2).

3.1.1 Security Analysis

Here we will prove that the proposed threshold RSA signature scheme is secure (i.e. existentially non-forgeable against an adaptive chosen message attack), provided that the RSA problem is intractable (i.e. RSA function is a one-way trapdoor function [18]). Throughout the thesis, we assume a static adversary model where the adversary controls exactly $t-1$ users and chooses them at the beginning of the attack. In this model, the adversary obtains all secret information of the corrupted users and the public parameters of the cryptosystem. She

can control the actions of the corrupted users, ask for partial signatures of the messages of her choice, but she cannot corrupt another user in the course of an attack, i.e., the adversary is static in that sense.

Theorem 3.1.1. *Given that the standard RSA signature scheme is secure, the threshold RSA signature scheme is secure under the static adversary model.*

Proof. To reduce the problem of breaking the standard RSA signature scheme to breaking the proposed threshold scheme, we will simulate the threshold protocol with no information on the secret where the output of the simulator is indistinguishable from the adversary's point of view. Afterwards, we will show that the secrecy of the private key d is not disrupted by the values obtained by the adversary. Thus, if the threshold RSA scheme is not secure, i.e., an adversary who controls $t-1$ users can forge signatures in the threshold scheme, one can use this simulator to forge a signature in the standard RSA scheme.

Let S' denote the set of users controlled by the adversary. To simulate the adversary's view, the simulator first selects a random interval $I = [a, b)$ from $\mathbb{Z}_M$, $M = \prod_{i=1}^{t} m_i$. The start point a is randomly chosen from $\mathbb{Z}_M$ and the end point is computed as $b = a + m_0 M_{S'}$. Then, the shares of the corrupted users are computed as $y_j = a \bmod m_j$ for $j \in S'$. Note that, these $t-1$ shares are indistinguishable from random ones due to (2.2) and the improved perfectness condition. Although the simulator does not know the real value of d, it is guaranteed that there exists a $y \in I$ which is congruent to $y_j \pmod{m_j}$ and $d \pmod{m_0}$ for all possible d values.

Since we have a (t, n)-threshold scheme, given a valid RSA signature (s, w), the partial signature s_i for a user $i \notin S'$ can be obtained by

$$s_i = s\kappa^{-\delta_S} \prod_{j \in S'} (w^{u_j})^{-1} \bmod N$$

where $S = S' \cup \{i\}$, $\kappa = w^{-M_S} \bmod N$ and δ_S is equal to either

$$\left\lfloor \frac{\sum_{j \in S'} u_j}{M_S} \right\rfloor + 1 \text{ or } \left\lfloor \frac{\sum_{j \in S'} u_j}{M_S} \right\rfloor.$$

The value of δ_S is important because it carries information on y. Let $U = \sum_{j \in S'} u_j$ and $U_S = U \mod M_S$. One can find whether y is greater than U_S or not by looking at δ_S:

$$\begin{aligned} y < U_S \quad &\text{if} \quad \delta_S = \lfloor U/M_S \rfloor + 1, \\ y \geq U_S \quad &\text{if} \quad \delta_S = \lfloor U/M_S \rfloor, \end{aligned}$$

Since the simulator does not know the real value of y, to determine the value of δ_S, the simulator acts according to the interval randomly chosen at the beginning of the simulation.

$$\delta_S = \begin{cases} \lfloor U/M_S \rfloor + 1, & \text{if} \quad a < U_S \\ \lfloor U/M_S \rfloor, & \text{if} \quad a \geq U_S \end{cases} \tag{3.3}$$

It is obvious that, the value of δ_S is indistinguishable from the real case if $U_S \notin I$. Now, we will prove that the δ_S values computed by the simulator does not disrupt the indistinguishability from the adversary's point of view. First of all, there are $(n-t+1)$ possible δ_S computed by using U_S since all the operations in the exponent depend on the coalition S alone. If none of the U_S values lies in I, the δ_S values observed by the adversary will be indistinguishable from a real execution of the protocol. Using this observation, we can prove that no information about the private key is obtained by the adversary.

Observing the $t-1$ randomly generated shares, there are $m_0 = \phi(N)$ candidates in I for y which satisfy $y_j = y \mod m_j$ for all $j \in S'$. These m_0 candidates have all different remainders modulo m_0 since $\gcd(M_{S'}, m_0) = 1$. So, exactly one of the remainders is equal to the private key d. If $U_S \notin I$ for all S, given an s_i, the shared value y can be equal to any of these m_0 candidates hence any two different values of the secret key d will be indistinguishable from adversary's point of view. In our case, this happens with all but negligible probability. First, observe that $U_S \equiv 0 \mod m_i$ and there are $m_0 M_{S'}/m_i$ multiples of m_i in I. Thus, the probability of $U_S \notin I$ for a coalition S is equal to

$$\left(1 - \frac{m_0 M_{S'}/m_i}{M_{S'}}\right) = \left(1 - \frac{m_0 M_{S'}}{M_S}\right).$$

According to (2.2), $m_i > {m_0}^2$ for all i hence the probability of $U_S \notin I$ for all possible S is less than $\left(1 - \frac{1}{m_0}\right)^{n-t+1}$, which is almost surely 1 for $m_0 \gg n$.

Consequently, the output of the simulator is indistinguishable from a real instance from the adversary's point of view, and hence the simulator can be used to forge a signature in the standard RSA scheme if the threshold RSA scheme can be broken. □

3.2 Using Chinese Remainder Theorem for Sharing Other Functions

3.2.1 Sharing of the ElGamal Decryption Function

The ElGamal cryptosystem [26] is another popular public key scheme proposed by T. ElGamal in 1989. It is an inherently probabilistic and semantically secure encryption scheme. For a cryptosystem to be semantically secure, it must be infeasible for a computationally-bounded adversary to derive significant information about a message (plaintext) when given only its ciphertext and the corresponding public encryption key. The description of the cryptosystem is given in Fig. 3.2.

ElGamal encryption scheme, like RSA, has the following multiplicative homomorphic property:

$$E(w) \times E(w') = E(ww')$$

for messages w and w' where E stands for the encryption function and $\times$ is the component-wise multiplication. Since the standard RSA encryption is deterministic, it is not semantically secure. One can use random padding to add semantic security as in [6]. However, this removes the homomorphic property. ElGamal does not suffer from such a problem since it is inherently semantically secure. This property makes ElGamal encryption suitable for use in threshold password authenticated key exchange protocols [1].

- *Setup*: Let p be a large prime and g be a generator of $\mathbb{Z}_p$. Choose a random $\alpha \in \{1, \ldots, p-1\}$ and compute $\beta = g^\alpha \bmod p$. (β, g, p) and α are the public and private keys, respectively.
- *Encryption*: Given a message $w \in \mathbb{Z}_p$, the ciphertext $c = (c_1, c_2)$ is computed as

$$c_1 = g^r \bmod p$$
$$c_2 = \beta^r w \bmod p$$

  where r is a random integer from $\mathbb{Z}_p$.
- *Decryption*: Given a ciphertext c, the message w is computed as

$$w = ({c_1}^\alpha)^{-1} c_2 \bmod p.$$

Figure 3.2: ElGamal's encryption scheme.

Threshold ElGamal Encryption Scheme: The following is a procedure that shares the ElGamal decryption function among n users with the Asmuth-Bloom SSS such that when t users come together they can decrypt the ciphertext:

- *Setup*: In the ElGamal setup phase, choose $p = 2q + 1$ where q is a large random prime and let $g \in \mathbb{Z}_p^*$ with order q. Choose a random $\alpha \in \{1, \ldots, p-1\}$ and compute $\beta = g^\alpha \bmod p$. Let α and (β, g, p) be the private and the public keys, respectively. Use Asmuth-Bloom SSS for sharing the private key α with $m_0 = 2q$.
- *Encryption* is the same as the standard ElGamal encryption.
- *Decryption*: Let (c_1, c_2) be the ciphertext to be decrypted where $c_1 = g^k \bmod p$ for some $k \in \{1, \ldots, p-1\}$ and $c_2 = \beta^k w$ where w is the message. The coalition S of t users wants to obtain the message $w = sc_2 \bmod p$ for the *decryptor* $s = (c_1^\alpha)^{-1} \bmod p$.

– *Generating partial results*: Each user $i \in S$ computes

$$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S, \tag{3.4}$$

$$s_i = {c_1}^{-u_i} \bmod p,$$

$$\beta_i = g^{u_i} \bmod p. \tag{3.5}$$

– *Combining partial results*: The incomplete decryptor $\overline{s}$ is obtained by combining the s_i values

$$\overline{s} = \prod_{i \in S} s_i \bmod p.$$

– *Correction*: The β_i values will be used to find the exponent which will be used to correct the incomplete decryptor. Compute the incomplete public key $\overline{\beta}$ as

$$\overline{\beta} = \prod_{i \in S} \beta_i \bmod p. \tag{3.6}$$

Let $\kappa_s = {c_1}^{M_S} \bmod p$ and $\kappa_\beta = g^{-M_S} \bmod p$ be the *correctors* for s and β, respectively. The corrector exponent δ can be obtained by trying

$$\overline{\beta}\kappa_\beta^j \stackrel{?}{\equiv} \beta \pmod p \tag{3.7}$$

for $0 \leq j < t$.

– *Extracting the message*: Compute the message w as

$$s = \overline{s}{\kappa_s}^{\delta} \bmod p,$$

$$w = sc_2 \bmod p.$$

where δ denotes the value of j that satisfies (3.7).

As in the case of RSA, the decryptor $\overline{s}$ is *incomplete* since we need to obtain $y = \sum_{i \in S} u_i \bmod M_S$ as the exponent of c_1^{-1}. Once this is achieved, $(c_1^{-1})^y \equiv (c_1^{-1})^\alpha \pmod p$ since $y = \alpha + A\phi(p)$ for some A.

When the equality in (3.7) holds we know that $\beta = g^\alpha \bmod p$ is the correct public key. This equality must hold for one j value, denoted by δ, in the given interval because since the u_i values in (3.4) and (3.5) are first reduced modulo

M_S. So, combining t of them will give $\alpha + am_0 + \delta M_S$ in the exponent in (3.6) for some $\delta \leq t-1$. Thus in (3.6), we obtained

$$\overline{\beta} = g^{\alpha+am_0+\delta M_S} \bmod p \equiv g^{\alpha+\delta M_S} = \beta g^{\delta M_S} = \beta \kappa_\beta^{-\delta} \pmod{p}$$

and for $j = \delta$ equality must hold. Actually, in (3.6) and (3.7), our purpose is not computing the public key since it is already known. We want to find the corrector exponent δ to obtain s, which is also equal to the one we use to obtain β. The equality can be verified as seen below:

$$\begin{aligned} s \equiv {c_1}^{-\alpha} &= \beta^{-r} \\ &= \left(g^{-(\alpha+(\delta-\delta)M_{\mathcal{S}})}\right)^r \\ &= {c_1}^{-(\alpha+am_0+\delta M_S)} \left({c_1}^{M_S}\right)^{\delta} = \overline{s}{\kappa_s}^{\delta} \pmod{p} \end{aligned}$$

3.2.1.1 Security Analysis

Here, we will prove that the threshold ElGamal encryption scheme is semantically secure provided that the standard ElGamal encryption scheme is semantically secure. We refer the reader to [29] for a formal definition of the threshold semantic security.

Theorem 3.2.1. *Given that the standard ElGamal encryption scheme is semantically secure, the threshold ElGamal encryption scheme is semantically secure under the static adversary model.*

Proof. The structure of the proof is similar to that we did for the threshold RSA signature scheme. Let S' denote the set of users controlled by the adversary. To simulate the adversary's view, the simulator first selects a random interval $I = [a, b)$ from $\mathbb{Z}_M$, $M = \prod_{i=1}^{t} m_i$. The start point a is randomly chosen from $\mathbb{Z}_M$ and the end point is computed as $b = a + m_0 M_{S'}$. Then, the shares of the corrupted users are computed as $y_j = a \bmod m_j$ for $j \in S'$.

Since we have a (t, n)-threshold scheme, when we determine the y_j values for $j \in S'$, the shares of other users are also determined. Although they cannot

be computed easily, given a valid message-ciphertext pair $(w, (c_1, c_2))$ the partial decryptor s_i and β_i for a user $i \notin S'$ can be obtained by

$$s_i = \left(wc_2^{-1}\right) \kappa_s{}^{-\delta_S} \prod_{j \in S'} c_1{}^{u_j} \bmod p,$$
$$\beta_i = \beta \kappa_\beta{}^{-\delta_S} \prod_{j \in S'} (\beta^{u_j})^{-1} \bmod p.$$

where $S = S' \cup \{i\}$, $\kappa_s = c_1{}^{M_S} \bmod p$, $\kappa_\beta = g^{-M_S} \bmod p$ and δ_S is equal to either

$$\left\lfloor \frac{\sum_{j \in S'} u_j}{M_S} \right\rfloor + 1 \text{ or } \left\lfloor \frac{\sum_{j \in S'} u_j}{M_S} \right\rfloor.$$

We use the same ideas to choose the value of δ_S as in the previous simulator so we skip the details and the analysis for the secrecy of the private key in the proof.

Consequently, the output of the simulator is indistinguishable from the adversary's point of view, and hence we proved that the threshold ElGamal scheme must be semantically secure if the standard one is. □

3.2.2 Sharing of the Paillier Decryption Function

Paillier's probabilistic cryptosystem [57] is a member of a different class of cryptosystems where the message is used in the exponent of the encryption operation. The description of the cryptosystem is given in Fig. 3.3.

Paillier's encryption scheme is probabilistic and has interesting homomorphic properties:

$$E(w_1)E(w_2) = E(w_1 + w_2)$$
$$E(w)^a = E(aw)$$

for messages, w, w_1, w_2 and a random integer a where E stands for the encryption function. These homomorphic properties make this encryption scheme suitable for different applications such as secure voting and lottery protocols [4, 29], DSA sharing protocols [51], and private information retrieval [56].

- *Setup*: Let $N = pq$ be the product of two large primes and $\lambda = \text{lcm}(p-1, q-1)$. Choose a random $g \in \mathbb{Z}_{N^2}$ such that the order of g is a multiple of N. The public and private keys are (N, g) and λ, respectively.
- *Encryption*: Given a message $w \in \mathbb{Z}_N$, the ciphertext c is computed as

$$c = g^w r^N \bmod N^2$$

  where r is a random number from $\mathbb{Z}_N$.
- *Decryption*: Given a ciphertext $c \in \mathbb{Z}_{N^2}$, the message w is computed as

$$w = \frac{L\left(c^\lambda \bmod N^2\right)}{L\left(g^\lambda \bmod N^2\right)} \bmod N$$

  where $L(x) = \frac{x-1}{N}$, for $x \equiv 1 \pmod N$.

Figure 3.3: Paillier's encryption scheme.

Threshold Paillier Encryption Scheme: The following is a procedure that shares the Paillier decryption function among n users with the Asmuth-Bloom SSS such that when t users come together they can decrypt the ciphertext. The setup part below is inspired by [29]:

- *Setup*: In the Paillier setup phase, choose large primes $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are also large random primes and $\gcd(N, \phi(N)) = 1$ for $N = pq$. Let $g = (1+N)^a b^N \bmod N^2$ for random a and b from $\mathbb{Z}_N^*$. Compute $\theta = a\beta\lambda \bmod N$ for a random $\beta \in \mathbb{Z}_N^*$ where $\lambda = \text{lcm}(p-1, q-1)$ is the Carmichael number for N. Let (N, g, θ) and λ be the public and private keys, respectively . Use the Asmuth-Bloom SSS to share $\beta\lambda$ with $m_0 = N\lambda$.
- *Encryption* is the same as the standard Paillier encryption.
- *Decryption*: Let $c = g^w r^N \bmod N^2$ be the ciphertext to be decrypted for some random $r \in \mathbb{Z}_N^*$ where w is the message from $\mathbb{Z}_N$. Assume a coalition S of size t wants to obtain the message $w = \frac{L(c^{\beta\lambda} \bmod N^2)}{\theta} \bmod N$. We call $s = c^{\beta\lambda} \bmod N^2$ as the *decryptor*.

– *Generating partial results*: Each user $i \in S$ computes

$$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S,$$
$$s_i = c^{u_i} \bmod N^2,$$
$$\theta_i = g^{u_i} \bmod N^2.$$

– *Combining partial results*: The incomplete decryptor $\overline{s}$ is obtained by combining the s_i values

$$\overline{s} = \prod_{i \in S} s_i \bmod N^2.$$

– *Correction*: The θ_i values will be used to find the exponent which corrects the incomplete decryptor. Compute the incomplete $\overline{\theta}$ as

$$\overline{\theta} = \prod_{i \in S} \theta_i \bmod N^2. \tag{3.8}$$

Let $\kappa_s = c^{-M_S} \bmod N^2$ and $\kappa_\theta = g^{-M_S} \bmod N^2$ be the *correctors* for s and θ, respectively. The corrector exponent δ can be obtained by trying

$$\theta \stackrel{?}{=} L(\overline{\theta}\kappa_\theta^j \bmod N^2) \tag{3.9}$$

for $0 \leq j < t$. Note that, for wrong corrector exponents L is undefined.

– *Extracting the message*: Compute the message w as

$$s = \overline{s}{\kappa_s}^{\delta} \bmod N^2,$$
$$w = \frac{L(s)}{\theta} \bmod N.$$

where δ denotes the value for j that satisfies (3.9).

The decryptor $\overline{s}$ is *incomplete* and to find the corrector exponent we used a similar approach. When the equality in (3.9) holds we know that $\theta = a\beta\lambda \bmod N^2$ is the correct value. Also, this equality must hold for one j value, denoted by δ, in the given interval. Actually, in (3.8) and (3.9), our purpose is not computing θ since it is already known. We want to find the corrector exponent δ to obtain s, which is also equal to the one we used to obtain θ.

3.2.2.1 Security Analysis

Here, we will prove that the threshold Paillier encryption scheme is semantically secure provided that the standard Paillier encryption scheme is semantically secure.

Theorem 3.2.2. *Given that the standard Paillier encryption scheme is semantically secure, the threshold Paillier encryption scheme is semantically secure under the static adversary model.*

Proof. The structure of the proof is similar to those we did for the previous threshold schemes. Let S' denote the set of users controlled by the adversary. To simulate the adversary's view, the simulator first selects a random interval $I = [a, b)$ from $\mathbb{Z}_M$, $M = \prod_{i=1}^{t} m_i$. The start point a is randomly chosen from $\mathbb{Z}_M$ and the end point is computed as $b = a + m_0 M_{S'}$. Then, the shares of the corrupted users are computed as $y_j = a \bmod m_j$ for $j \in S'$.

Since we have a (t, n)-threshold scheme, when we determine the y_j values for $j \in S'$, the shares of other users are also determined. Although they cannot be computed easily, given a valid message-ciphertext pair (w, c) the decryptor share s_i and θ_i for a user $i \notin S'$ can be obtained by

$$s_i = (1 + w\theta N){\kappa_s}^{-\delta_S} \prod_{j \in S'} ({c_1}^{u_j})^{-1} \bmod N^2,$$
$$\theta_i = (1 + \theta N){\kappa_\theta}^{-\delta_S} \prod_{j \in S'} (\theta^{u_j})^{-1} \bmod N^2.$$

where $S = S' \cup \{i\}$, $\kappa_s = c^{-M_S} \bmod N^2$, $\kappa_\theta = g^{-M_S} \bmod N^2$ and δ_S is equal to either

$$\left\lfloor \frac{\sum_{j \in S'} u_j}{M_S} \right\rfloor + 1 \text{ or } \left\lfloor \frac{\sum_{j \in S'} u_j}{M_S} \right\rfloor.$$

We use the same ideas to choose the value of δ_S as in the previous simulator so we skip the details and the analysis for the secrecy of the private key in the proof.

Consequently, the output of the simulator is indistinguishable from the adversary's point of view, and hence we proved that the threshold Paillier scheme must be semantically secure if the standard one is. □

3.2.3 Sharing of the Naccache-Stern Decryption Function

A different cryptosystem which uses bitwise encryption was proposed by Naccache and Stern [54]. This cryptosystem is based on a type of knapsack problem: Given arbitrary integers c, l, p, and a vector of integers $x = (x_1, ..., x_n)$, find a vector $w \in \{0,1\}^l$ such that

$$c \equiv \prod_{i=1}^{l} {x_i}^{w_i} \bmod p \tag{3.10}$$

When the x_i are relatively prime and much smaller than the modulus p, this knapsack problem can be solved easily. When x_i are arbitrary numbers in $\mathbb{Z}_p$, the problem is hard. The cryptosystem is given in Figure 3.4.

- *Setup*: Let p be a large prime, l be a positive integer and for i from 1 to l, set p_i to be the ith prime, starting with $p_1 = 2$. Choose a secret integer $d < p-1$, such that $\gcd(p-1, d) = 1$. Set $v_i = \sqrt[d]{p_i} \bmod p$. The public key is then p, l, $v = (v_1, \ldots, v_l)$. The private key is d.
- *Encryption*: To encrypt an l-bit long message w, calculate

$$c = \prod_{i=1}^{l} {v_i}^{w_i} \bmod p. \tag{3.11}$$

  where w_i is the ith bit of message w.
- *Decryption*: One can obtain the plaintext by computing

$$w = \sum_{i=1}^{l} \frac{\gcd(p_i, c^d \bmod p) - 1}{p_i - 1} \times 2^i. \tag{3.12}$$

Figure 3.4: Naccache-Stern's encryption scheme.

Threshold Naccache-Stern Encryption Scheme: To the best of our knowledge, no FSSs have been proposed for the Naccache-Stern knapsack cryptosystem. Here we give the first realization of an FSS for this cryptosystem with Asmuth-Bloom SSS:

1. In the Naccache-Stern Knapsack setup, choose p be a safe prime, l be a positive integer and for i from 1 to l, set p_i to be the ith prime, starting with $p_1 = 2$. Choose a secret integer $d < p-1$, such that $\gcd(p-1, d) = 1$. Set $x_i = \sqrt[d]{p_i} \bmod p$. Set the public key be p, l, x. The private key d is shared with $m_0 = p - 1$.

2. Let c be the ciphertext to be decrypted where $c = \prod_{i=1}^{l} {x_i}^{w_i} \bmod p$ and assume a coalition S of size t wants to obtain the plaintext w. The ith person in the coalition knows m_j for all $j \in S$ and $y_i = y \bmod m_i$ as its secret share.

3. Each user $i \in S$ computes

$$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_{\mathcal{S}},$$
$$s_i = c^{u_i} \bmod p.$$

4. The incomplete decryptor $\overline{s}$ is obtained by combining the s_i values

$$\overline{s} = \prod_{i \in S} s_i \bmod p. \tag{3.13}$$

5. Let $\kappa = c^{-M_S} \bmod p$ be the *corrector*. The corrector exponent δ can be obtained by trying

$${x_1}^{\overline{s}\kappa^j} \stackrel{?}{\equiv} 2 \bmod p \tag{3.14}$$

for $0 \leq j < t$.

6. Compute the plaintext message w as

$$s = \overline{s}\kappa^{\delta} \bmod p,$$
$$w = \sum_{i=1}^{l} \frac{(\gcd(p_i, s \bmod p) - 1)}{p_i - 1} \times 2^i.$$

Where δ denotes the j value that satisfies (3.14).

The decryptor $\overline{s}$ is *incomplete* since we need to obtain $y = \sum_{i \in S} u_i \bmod M_S$ as the exponent of c. Once this is achieved, $c^y \equiv c^d \bmod p$, since $y = d + a(p-1)$ for some a.

Note that the equality in (3.14) must hold for one $j \leq t-1$ since the u_i values were already reduced modulo M_S. So, combining t of them in (3.13) will give $d + am_0 + \delta M_S$ in the exponent for some $\delta \leq t-1$. Thus we obtained

$$\overline{s} = c^{d+am_0+\delta M_S} \equiv c^{d+\delta M_S} \equiv sc^{\delta M_S} \equiv s\kappa^{-\delta} \bmod p \tag{3.15}$$

and for $j = \delta$, equation (3.14) will hold.

3.3 Efficiency Analysis of the Proposed Schemes

Although the proposed schemes are not more efficient than Shoup's work [68], which is the fastest threshold RSA signature scheme, they are comparable in performance. In this section, we give an efficiency analysis of the proposed schemes. First, we compare the proposed threshold RSA scheme with the basic RSA scheme in [68] in terms of share size and computation cost. For the computation cost, the dominating factor is the exponentiation operations hence we are mainly interested in the number of exponentiations. Note that, the cost of an exponentiation is proportional to the size of the exponent.

- *Share size:* In [68], the size of a share is approximately k bits for a k-bit modulus N. In our case, because of (2.2) the size of a share is about $2k$ bits for the same N.
- *Computing partial signatures:* In [68], it takes an exponentiation with a $(k + \log(n!))$-bit exponent to compute a partial signature. In the proposed scheme,

$$u_i = y_i M'_{S,i} M_{S \setminus \{i\}} \bmod M_S$$

is a $2kt$-bit integer. To compute it efficiently we first compute $M'_{S,i}$ and $r = \lfloor y_i M'_{S,i}/m_i \rfloor$ which are $2k$-bit integers. Now u_i is equal to

$$u_i = M_{S\setminus\{i\}}(y_i M'_{S,i} - r m_i)$$

and computing the partial signature $s_i = w^{u_i} \bmod N$ needs a modular exponentiation with $2kt$-bit exponent. Note that no extra storage is needed to store u_i.

- *Combining partial signatures:* In [68], combining the partial results requires t exponentiations with approximately $\log(n!)$-bit exponents, hence the cost is $t\log(n!)$. After that, these t results are multiplied to obtain the signature. In the proposed scheme, after obtaining the incomplete signature, an exponentiation with a $2kt$-bit exponent is needed to compute the corrector. Note that while computing the partial signature the ith player computes $w^{M_{S\setminus\{i\}}} \bmod N$ as an intermediate value. The combiner can compute its inverse and raise it to the m_ith power to compute the corrector which requires an exponentiation with $2k$-bit exponent rather than $2kt$. After that, at most $2t$ more multiplications are required for computing the incomplete signature and checking equation (3.2).

Criteria	Shoup's scheme	Proposed scheme
Share sizes	k	$2k$
Cost of computing partial signatures	$k + \log(n!)$	$2kt$
Cost of combining partial signatures	$t\log(n!)$	$2k$

Table 3.1: Comparison of the proposed threshold RSA signature scheme with Shoup's scheme [68] in terms of the share sizes, and the cost of computing and combining the partial signatures measured in terms of the total size of exponents.

Table 3.1 compares the performance of the proposed scheme with that of [68]. Although not more efficient, the proposed RSA signature scheme is comparable in performance to Shoup's scheme given that t is a small integer, which is the case in a typical application. Regarding the proposed threshold ElGamal and Pallier schemes, their complexities are similar to that of the threshold RSA scheme and hence the comparisons are similar to that in Table 3.1.

Chapter 4

Sharing DSS with CRT

The Digital Signature Standard (DSS) is the current U.S. standard for digital signatures. Sharing DSS is an interesting problem and a neat solution was given by Gennaro et al. [34] based on Shamir's SSS. In this chapter, we propose a new threshold scheme for the Digital Signature Standard by using the Asmuth-Bloom SSS. To the best of our knowledge, this is the first provably secure threshold DSS scheme based on the Chinese Remainder Theorem. The DSS scheme is given in Fig. 4.1.

4.1 Modifications on Asmuth-Bloom SSS for DSS

To adapt the original scheme for threshold DSS, if $n > 3t-1$, we first modify the equation (2.1) used in the Asmuth-Bloom secret sharing scheme in Chapter 2 as

$$\prod_{i=1}^{2t} m_i > n{m_0}^2 \prod_{i=1}^{2t-1} m_{n-i+1}. \tag{4.1}$$

Note that if $n > 3t-1$, (4.1) also implies that

$$\prod_{i=1}^{t} m_i > n{m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{4.2}$$

- *Key Generation Phase:* Let p and q be large prime numbers where $q|p-1$ and $g \in \mathbb{Z}_p^*$ be an element of order q. The private key $\alpha \in_R \mathbb{Z}_q^*$ is chosen randomly and the public key $\beta = g^\alpha \bmod p$ is computed.
- *Signing Phase:* The signer first chooses a random ephemeral key $k \in_R \mathbb{Z}_q^*$ and then computes the signature (r, s) where

$$r = (g^{k^{-1}} \bmod p) \bmod q$$
$$s = k(w + \alpha r) \bmod q$$

  for a hashed message $w \in \mathbb{Z}_q$.
- *Verification Phase:* The signature (r, s) is verified by checking

$$r \stackrel{?}{=} (g^{ws^{-1}} \beta^{rs^{-1}} \bmod p) \bmod q$$

  where s^{-1} is computed in $\mathbb{Z}_q^*$.

Figure 4.1: The DSS scheme.

If $n \leq 3t-1$, (4.2) implies (4.1) and we generate the m_is with respect to (4.2). In general, we will use (4.2) for t-out-of-n secret sharing schemes in the primitives which will be described later. For one primitive, JOINT-ZS, which will be described in next section, we will use a $2t$-out-of-n sharing by using (4.1). We also change the definition of M as

$$M = \left\lfloor \frac{\prod_{i=1}^{t} m_i}{n} \right\rfloor .$$

4.1.1 Arithmetic Properties of the Modified Asmuth-Bloom SSS

Suppose multiple secrets are shared with common parameters t, n, and moduli m_is. The shareholders can use the following properties to obtain new shares for the sum and product of the shared secrets.

Proposition 4.1.1. *Let $d_1, d_2, \cdots, d_n$ be secrets shared by Asmuth-Bloom SSS with common parameters t, n, and moduli m_is. Let y_{ij} be the share of the ith*

user for secret d_j. Then, for

$$D = (\sum_{i=1}^{n} d_i) \bmod m_0$$
$$Y_i = (\sum_{j=1}^{n} y_{ij}) \bmod m_i,$$

we have

$$D \overset{t}{\leftrightarrow} (Y_1, Y_2, \cdots, Y_n),$$

i.e., the secret D is t-out-of-n shared with shares $Y_1, Y_2, \cdots, Y_n$.

Proof. For

$$Y = \sum_{i=1}^{n} (d_i + A_i m_0),$$

we have $Y_i \equiv Y \bmod m_i$. Note that due to (4.2), $Y < nM < M_S$ for any coalition S where $|S| \geq t$. Hence, a coalition S of t users can construct $Y \in M_S$ and obtain $D = Y \bmod m_0$. □

Proposition 4.1.2. *Let d_1, d_2 be secrets shared by Asmuth-Bloom SSS with common parameters t, n and moduli m_is. Let y_{ij} be the share of the ith user for secret d_j. Then, for*

$$D = d_1 d_2 \bmod m_0,$$
$$Y_i = y_1 y_2 \bmod m_i,$$

we have

$$D \overset{2t}{\leftrightarrow} (Y_1, Y_2, \cdots, Y_n).$$

Proof. For

$$Y = \prod_{i=1}^{2} (d_i + A_i m_0),$$

we have $Y_i \equiv Y \bmod m_i$. Note that $Y < M^2 < M_S$ for any coalition S where $|S| \geq 2t$. Hence, a coalition S of $2t$ users can construct $Y \in M_S$ and obtain $D = Y \bmod m_0$. □

4.2 The Threshold DSS Scheme

To obtain a threshold DSS scheme, first the dealer generates the private key α and shares it among the users by a (t, n) Asmuth-Bloom secret sharing scheme with $m_0 = q$. Then a signing coalition S can sign a message in a threshold fashion without requiring a trusted party. Note that anyone can obtain the secret key α and forge signatures if he knows k for a valid signature (r, s). Hence, $r = (g^{k^{-1}} \bmod p) \bmod q$ must be computed in a way that no one obtains k. Here, we first explain the necessary primitives that will be used to solve this problem and then describe the overall threshold signature scheme together. Below, S denotes the signing coalition of size $2t+1$. Without loss of generality, we assume $S = \{1, 2, \ldots, 2t+1\}$. We will first describe the primitive tools we used in the proposed CRT-based threshold DSS scheme. In these primitives, we set $m_0 = q$.

4.2.1 Joint Random Secret Sharing

In a joint random secret sharing scheme (JOINT-RSS), each user in the signing coalition S contributes something to the secret generation process and obtains a share for the resulting random secret as described in Fig. 4.2. A verifiable version of this scheme can be found in Chapter 5.

- Each user $j \in S$ chooses a random secret $d_j \in \mathbb{Z}_{m_0}$ and shares it as $d_j \overset{t}{\hookrightarrow} (y_{1j}, y_{2j}, \cdots, y_{(2t+1)j})$ where y_{ij} is the share of the ith user.
- The ith user computes
$$Y_i = \left(\sum_{j=1}^{2t+1} y_{ij} \right) \bmod m_i.$$
By Proposition 4.1.1, $D \overset{t}{\hookrightarrow} (Y_1, Y_2, \ldots, Y_{2t+1})$ is a valid and secure t-out-of-$(2t+1)$ SSS for
$$D = \left(\sum_{i=1}^{2t+1} d_i \right) \bmod m_0.$$

Figure 4.2: CRT-based JOINT-RSS procedure.

4.2.2 Joint Zero Sharing

In a joint zero sharing scheme (JOINT-ZS), each user in the signing coalition S contributes something to the random zero generation process and obtains a share for the resulting zero, as described in Fig. 4.3.

- Each user $j \in S$ shares $0 \overset{2t}{\leftrightarrow} (y_{1j}, y_{2j}, \cdots, y_{(2t+1)j})$ where $y_{ij} = A_j m_0 \bmod m_i$ is the share of the ith user for some $A_j m_0 < M$. Here the value M is modified as follows:

$$M = \left\lfloor \frac{\prod_{i=1}^{2t} m_i}{n} \right\rfloor .$$

  Note that the moduli m_is satisfy the equation (4.1).

- The ith user computes

$$Y_i = \left(\sum_{j=1}^{2t+1} y_{ij} \right) \bmod m_i.$$

  By Proposition 4.1.1, $D \overset{2t}{\leftrightarrow} (Y_1, Y_2, \ldots, Y_{2t+1})$ is a valid $2t$-out-of-$(2t+1)$ scheme.

Figure 4.3: CRT-based JOINT-ZS procedure.

4.2.3 Computing $g^d \bmod p$

For threshold DSS, we will need to share and compute $g^d \bmod p$ for a joint random secret $d \in \mathbb{Z}_q$. Fig. 4.4 describes a scheme, JOINT-EXP-RSS, to construct an approximate value for $F_d = g^d \bmod p$. This approximate value will later be corrected through a separate correction process.

Observe that $d = ((\sum_{i \in S} u_i) \bmod M_S) \bmod q$ whereas this construction process computes $F_{d'} = g^{d'} \bmod p$ for $d' = \sum_{i \in S} u_i \bmod q$. Since there are $2t+1$ users in S and $u_i < M_S$ for all i, $d = d' - \delta_d M_S \bmod q$ for some integer $0 \leq \delta_d \leq 2t$.

- To compute $F_d = g^d \bmod p$ for a random secret value d, S uses JOINT-RSS to generate and share d as $d \overset{t}{\leftrightarrow} (y_1, y_2, \ldots, y_{2t+1})$ with $m_0 = q$.
- Each user $i \in S$ computes

$$u_{i,d} = (y_i M_{S\setminus\{i\}} M'_{S,i}) \bmod M_S$$

where $M'_{S,i}$ is the inverse of $M_{S\setminus\{i\}} \bmod m_i$, and broadcasts

$$f_{i,d} = g^{u_{i,d}} \bmod p.$$

- The approximate value for $g^d \bmod p$ is computed as

$$F_{d'} = \prod_{i \in S} f_{i,d} \bmod p.$$

Figure 4.4: CRT-based JOINT-EXP-RSS procedure.

4.2.4 Computing $g^{k^{-1}} \bmod p$

In DSS, we need to compute $r = g^{k^{-1}} \bmod p$ in such a way that neither k nor k^{-1} is known by any user. The JOINT-EXP-INVERSE procedure described in Fig. 4.5 computes r without revealing k.

Note that the (j_a, j_k) pair, $0 \leq j_a, j_k \leq 2t$, found for (4.3) is unique with overwhelming probability given that $(2t+1)^2 \ll q$.

4.2.5 The Overall Scheme

The phases of the proposed threshold DSS scheme are described below:

- *Key Generation Phase:* Let $\alpha \in_R \mathbb{Z}_q^*$ be the private signature key. The dealer sets $m_0 = q$ and shares $\alpha \overset{t}{\leftrightarrow} (\alpha_1, \alpha_2, \ldots, \alpha_n)$.
- *Signing Phase:* To sign a hashed message $w \in \mathbb{Z}_q$, the signing coalition S of size $2t+1$ first computes

$$r = (g^{k^{-1}} \bmod p) \bmod q$$

- S uses JOINT-RSS to jointly share random secrets $k \overset{t}{\leftrightarrow} (k_1, k_2, \ldots, k_{2t+1})$, $a \overset{t}{\leftrightarrow} (a_1, a_2, \ldots, a_{2t+1})$, and uses JOINT-ZS to distribute shares for zero, i.e., $0 \overset{2t}{\leftrightarrow} (z_1, z_2, \ldots, z_{2t+1})$.
- S constructs $v = ak$ from shares $v_i = (a_i k_i + z_i) \bmod m_i$, $i \in S$. Note that $v \overset{2t+1}{\leftrightarrow} (v_1, v_2, \ldots, v_{2t+1})$ by Propositions 4.1.1 and 4.1.2.
- S uses JOINT-EXP-RSS to obtain

$$F_{a'} = \prod_{i \in S} f_{i,a} = \prod_{i \in S} g^{u_{i,a}} \equiv g^{a'} \equiv g^{a+\delta_a M_S} \bmod p,$$
$$F_{k'} = \prod_{i \in S} f_{i,k} = \prod_{i \in S} g^{u_{i,k}} \equiv g^{k'} \equiv g^{k+\delta_k M_S} \bmod p.$$

  S also computes

$$F_{a'k'} = \prod_{i \in S} f_{i,ak} = \prod_{i \in S} {F_{a'}}^{u_{i,k}} \equiv g^{a'k'} \equiv g^{(a+\delta_a M_S)(k+\delta_k M_S)} \bmod p$$
$$\equiv g^v {F_{a'}}^{\delta_k M_S} {F_{k'}}^{\delta_a M_S} g^{-\delta_a \delta_k M_S^2} \bmod p.$$

- S checks the following equality for all $0 \le j_a, j_k \le 2t$

$$F_{a'k'} \overset{?}{=} g^v {F_{a'}}^{j_k M_S} {F_{k'}}^{j_a M_S} g^{-j_a j_k M_S^2} \bmod p \tag{4.3}$$

  and finds the $(j_a = \delta_a, j_k = \delta_k)$ pair that satisfies this equality. Once δ_a is found

$$F_a = g^a \bmod p = F_{a'} g^{-\delta_a M_S} \bmod p$$

  can be computed.
- The signing coalition S computes

$$g^{k^{-1}} \bmod p = {F_a}^{(v^{-1})} \bmod p.$$

Figure 4.5: CRT-based JOINT-EXP-INVERSE procedure.

by JOINT-EXP-INVERSE described in Fig. 4.5. To compute

$$s = k(w + r\alpha) \bmod q,$$

each user $i \in S$ computes

$$s_i = k_i(w + r\alpha_i) \bmod m_i$$

and broadcasts it. Then the signature s is computed by using the reconstruction process for the Asmuth-Bloom SSS with $2t + 1$ shares.

- *Verification Phase* is the same as the standard DSS verification.

Since α is shared (t, n), the value $\alpha + A_\alpha m_0$ is less than M and, $r, w < q = m_0$. Hence,

$$w + ry < m_0 + m_0\,(\alpha + A_\alpha m_0) < (m_0 + 1)M$$

and a coalition of size $t+1$ is sufficient to compute $w+ry$ and obtain $w+r\alpha \bmod q$. Since the threshold for the secret k is t, by Proposition 4.1.2, $s \overset{2t+1}{\leftrightarrow} (s_1, s_2, \ldots, s_n)$ and s can be computed by $2t + 1$ partial signatures.

4.3 Security Analysis

Here we will prove that the proposed threshold DSS signature scheme is secure (i.e. existentially non-forgeable against an adaptive chosen message attack), provided that the DSS signatures are unforgeable. Throughout the paper, we assume a static adversary model where the adversary controls exactly $t - 1$ users and chooses them at the beginning of the attack. In this model, the adversary obtains all secret information of the corrupted users and the public parameters of the cryptosystem. She can control the actions of the corrupted users, ask for partial signatures of the messages of her choice, but she cannot corrupt another user in the course of an attack, i.e., the adversary is static in that sense.

First we recall that the modified Asmuth-Bloom scheme is perfect:

Theorem 4.3.1. *The modified secret sharing scheme with the new M and equation (4.2) is perfect in the sense that the probabilities* $\Pr(d = d')$ *and* $\Pr(d = d'')$ *are approximately equal for all* $d', d'' \in \mathbb{Z}_{m_0}$.

Proof. Let S' be a corrupted coalition of $t-1$ users. For perfectness, we need to check the value of $M/M_{S'}$ which is

$$\frac{M}{M_{S'}} > \frac{\prod_{i=1}^{t} m_i}{n \prod_{i=1}^{t-1} m_{n-i+1}} > {m_0}^2$$

due to equation (4.2). Similar to the proof of Theorem 2.2.1, we can see that the perfectness condition is preserved. □

To reduce the problem of breaking the DSS signature scheme to breaking the proposed threshold scheme, we will simulate the protocol with no information on the secret where the output of the simulator is indistinguishable from the adversary's point of view. The input to the simulator is the hashed message w, its signature (r, s), the public key β, the secret shares of the corrupted users, i.e., $\alpha_i \in S_B$, where S_B denotes the corrupted (bad) user set. Let S_G be the set of good users in S and let

$$r^* = g^{ms^{-1}} \beta^{rs^{-1}} \bmod p.$$

The actions of the simulator is described below:

1. By simulating the good users in S_G, with JOINT-RSS procedure, the simulator shares random values for each user in S_G. It also obtains the good users' shares from the corrupted users in S_B. Note that all of these values are known by the simulator since $|S_G| \geq t$, which is the threshold. Let $\overline{a}, \overline{k} \in \mathbb{Z}_q$ be the shared values in this step, i.e., $\overline{k} \overset{t}{\leftrightarrow} (\overline{k}_1, \overline{k}_2, \ldots, \overline{k}_{2t+1})$ and $\overline{a} \overset{t}{\leftrightarrow} (\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_{2t+1})$. After that, 0 is shared by using the procedure JOINT-ZS, i.e., $0 \overset{2t}{\leftrightarrow} (\overline{z}_1, \overline{z}_2, \ldots, \overline{z}_{2t+1})$. For the rest of the simulation, let $\delta_{\overline{a}} = \left\lfloor \frac{\sum_{i \in S} u_{i,\overline{a}}}{M_S} \right\rfloor$ and $\delta_{\overline{k}} = \left\lfloor \frac{\sum_{i \in S} u_{i,\overline{k}}}{M_S} \right\rfloor$.

2. By using the 2nd step in Fig. 4.5, $\overline{v} = \overline{a}\overline{k}$ is computed. Let $\overline{F_{a'}} = {r^*}^{\overline{v}} g^{\delta_{\overline{a}} M_S}$. The simulator uses $\overline{a}_i$ values to compute $\overline{f_{i,a}} = g^{\overline{a}_i M_{S \setminus i} M'_{S,i} \bmod M_S} \bmod p$ for

all $i \in S_G$ but one. For the last user, $\overline{f_{i,a}}$ is selected such that

$$\prod_{i \in S_G} \overline{f_{i,a}} \equiv \overline{F_{a'}} \left(\prod_{i \in S_B} \overline{f_{i,a}} \right)^{-1} \bmod p. \tag{4.4}$$

These $\overline{f_{i,a}}$ values are then broadcast.

3. By using the construction phase of JOINT-EXP-RSS, $F_{\overline{k'}} = \prod_{i \in S} f_{i,\overline{k}} \bmod p$ is computed. Let

$$\overline{F_{a'k'}} = \overline{F'_a}^{\delta_{\overline{k}} M_S} F_{\overline{k'}}^{\delta_{\overline{a}} M_S} g^{-\delta_{\overline{a}} \delta_{\overline{k}} M_S^2} g^{\overline{v}} \bmod p.$$

The simulator uses $\overline{k}_i$ values to compute $\overline{f_{i,ak}} = \overline{F_{a'}}^{\overline{k_i} M_{S \setminus i} M'_{S,i} \bmod M_S} \bmod p$ for all $i \in S_G$ but one. For the last user, $\overline{f_{i,ak}}$ is selected such that

$$\prod_{i \in S_G} \overline{f_{i,ak}} \equiv \overline{F_{a'k'}} \left(\prod_{i \in S_B} \overline{f_{i,ak}} \right)^{-1} \bmod p.$$

These $\overline{f_{i,ak}}$ values are then broadcasted. After that the correction phase is completed.

4. Let $\overline{s_i} = \overline{k}_i (w + \alpha_i r) \bmod q$ for $i \in S_B$. The simulator chooses a random integer $\overline{U_s}$ smaller than $M (m_0 + m_0 M)$ such that $\overline{U_s} \equiv \overline{s_i} \bmod m_i$ for $i \in S_B$ and $\overline{U_s} \equiv s \bmod m_0$. Then it computes $\overline{s_i} = \overline{U_s} \bmod m_i$ for $i \in S_G$ and broadcasts them. After these steps, the signature (r, s) is computed.

To prove that the outcome of the simulator is indistinguishable, we first need to state the following conjecture:

Conjecture 4.3.1 (Gennaro et al. [34])**.** Let G be the subgroup generated by g. Choose u, v at random, uniformly distributed and independently in $\mathbb{Z}_q$. The following probability distributions on $G \times G$,

$$(g^u \bmod p, g^v \bmod p) \text{ and } \left(g^u \bmod p, g^{u^{-1}} \bmod p \right)$$

are computationally indistinguishable.

A similar assumption is also used in [60] to prove the security of the proposed anonymous fingerprinting scheme. Also in [3], Bao et al. proved that if the Computational Diffie-Hellman (CDH) assumption holds there is no probabilistic polynomial time Turing machine which outputs $g^{x^{-1}}$ on inputs g and g^x with non-negligible probability. Note that the CDH assumption states that there is no probabilistic polynomial time Turing machine which outputs g^{xy} on inputs g, g^x and g^y with non-negligible probability. The decisional version of CDH is called as the Decisional Diffie-Hellman (DDH) assumption. DDH states that the following probability distributions on $G \times G \times G$,

$$(g^u, g^v, g^{uv}) \text{ and } (g^u, g^v, g^z)$$

are computationally indistinguishable where u, v, z are random, uniformly distributed and independent in $\mathbb{Z}_q$.

Lemma 4.3.2. *The outcome of the simulator is indistinguishable from the CRT-based threshold DSS signature from adversary's point view.*

Proof. The steps of the simulator are indistinguishable from the real execution of the protocol as proved below:

1. As shown in Theorem 4.3.1 the modified SSS is perfect, i.e., the probabilities $\Pr(d = \overline{k})$ and $\Pr(d = k)$ are approximately equal for $k, \overline{k} \in \mathbb{Z}_{m_0}$ where d is the shared secret, k is the shared value in real protocol and $\overline{k}$ is the shared value in the simulation. The same argument is also true for $\overline{a}$.

2. In the real protocol, the set of shares $(v_1, v_2, \ldots, v_{2t+1})$ is a valid sharing for a uniformly distributed value v. In the simulation, $(\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_{2t+1})$ also yields a uniformly distributed value $\overline{v}$. Hence, the distribution of the shares v_i, $i \in S$, is identical to the distribution of $\overline{v}_i$, $i \in S$. Note that, if the joint zero-sharing procedure is not used, i.e., if the shares of v are not randomized, the secrecy of a and k is not preserved.

   In the real protocol, $F_{a'} = g^{a+\delta_a M_S} \bmod p$ where a is uniformly random and δ_a is another random value independent from a. The simulation computes $\overline{F_{a'}} = g^{k^{-1}\overline{v}+\delta_{\overline{a}} M_S} \bmod p$. Since $\overline{v}$ is uniformly random, $k^{-1}\overline{v}$ is also uniformly

random. The simulator uses the exact $\delta_{\overline{a}}$ value determined in Step 2 so its distribution is identical to that of δ_a. Hence, the distribution of $F_{a'}$ and $\overline{F_{a'}}$ values are identical.

In the simulation, $\overline{a}_i$s are used to compute $\overline{f_{i,a}} = g^{\overline{a}_i M_{S\setminus i} M'_{S,i} \bmod M_S} \bmod p$, and in the computation of $f_{i,a} = g^{u_{i,a}}$, thanks to perfectness, the share a_i can be any integer from $\mathbb{Z}_q$. Hence, the distributions of $f_{i,a}$s and $\overline{f_{i,a}}$s are indistinguishable for the users in $S \setminus \{j\}$. However for the last user j of S_G, the simulator chooses a specific $\overline{f_{j,a}}$ to satisfy equation (4.4). We will show that without $\overline{f_{j,a}}$, the rest of the $\overline{f_{i,a}}$s for $i \in S \setminus \{j\}$ yield a random value in the group generated by g. Let $S' = S \setminus \{j\}$. Consider

$$\sum_{i \in S'} u_{i,\overline{a}} = \sum_{i \in S'} \overline{a}_i M_{S\setminus\{i\}} M'_{S,i} \bmod M_S.$$

Since the threshold for $\overline{a}$ is t and $|S'| > t$, the following equation is satisfied:

$$\overline{a} = \left(\left(\sum_{i \in S'} u_{i,\overline{a}} \right) \bmod M_{S'} \right) \bmod q.$$

However, when we try to do the same construction in the exponent,

$$\prod_{i \in S'} \overline{f_{i,a}} \equiv g^{\overline{a} + \Delta_{\overline{a}} M_{S'}} \bmod p$$

for some $\Delta_{\overline{a}} < |S'| \frac{M_S}{M_{S'}} = |S'| m_j$. Since, $\Delta_{\overline{a}}$ is an unknown, $m_j > {m_0}^2 = q^2$, and $\gcd(q, M_{S'}) = 1$, we have $g^{\overline{a} + \Delta_{\overline{a}} M_{S'}}$ uniformly random in the group generated by g. Note that this is true for every $S' \subset S$, i.e., there is no correlation between $\overline{f_{i,a}}$s for $i \in S'$ when $u_{i,\overline{a}}$s are computed for a larger coalition $S \supset S'$. Therefore, in the simulation, the adversary cannot distinguish the inconsistency of the last user's $\overline{f_{j,a}}$. Hence, the distributions of $\overline{f_{i,a}}$ and $f_{i,a}$ for $i \in S$ are indistinguishable.

The same argument is also true for the distributions of $\overline{f_{i,ak}}$ and $f_{i,ak}$ for $i \in S$ which are used in the following step.

3. For the correction phase in the real protocol, the values $f_{i,ak}$ and $f_{i,k}$ use the same value $u_{i,k}$ in the exponent, likewise, the ones in the simulator. The correction equation used in the real protocol is

$$F_{a'k'} = {F_{a'}}^{\delta_k M_S} {F_{k'}}^{\delta_a M_S} g^{-\delta_a \delta_k M_S^2} g^v \bmod p.$$

And, the simulator uses the value

$$\overline{F_{a'k'}} = \overline{F_{a'}}^{\delta_{\overline{k}} M_S} F_{\overline{k'}}^{\delta_{\overline{a}} M_S} g^{-\delta_{\overline{a}}\delta_{\overline{k}} M_S^2} g^{\overline{v}} \bmod p$$

where the distribution of each value on the right side is identical to that of the corresponding value in the real protocol. Hence the distribution of $F_{a'k'}$ and $\overline{F_{a'k'}}$ values are identical. The distributions of the outputs of the correction processes, i.e., δ_a and δ_k, are also identical since the simulator uses actual the $\delta_{\overline{a}}$ and $\delta_{\overline{k}}$ values.

Here we need to use the DDH assumption and Conjecture 4.3.1. After the correction, $\overline{F_{a'}}$, $F_{\overline{k'}}$ and $\overline{F_{a'k'}}$ are revealed where in the real protocol they are equal to $g^{a'}$, $g^{k'}$ and $g^{a'k'}$. The DDH assumption states that the distributions of these triplets are indistinguishable. Besides, $F_{\overline{k'}} = g^{\overline{k'}}$ and, once $\delta_{\overline{k}}$ is found, $g^{\overline{k}}$ can be computed. The users will also know $r = g^{k^{-1}}$ and in the real protocol the pair $(g^{\overline{k}}, g^{k^{-1}})$ will be $(g^k, g^{k^{-1}})$. Conjecture 4.3.1 says that the distributions of these two pairs are also indistinguishable.

4. In the real protocol, the set of shares $(s_1, s_2, \ldots, s_{2t+1})$ is a valid sharing for a uniformly distributed value s. In the simulation, $(\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_{2t+1})$ also yields the same value. The computing process of $\overline{s}_i$ for $i \in S_G$ is the same as the one in the real protocol. Hence, the distribution of the shares s_i, $i \in S$, is identical to the distribution of $\overline{s}_i$, $i \in S$.

□

We conclude this chapter with the following theorem, which is a corollary to Theorem 4.3.1 and Lemma 4.3.2.

Corollary. *Given that the standard DSS signature scheme is secure, the threshold DSS signature scheme is also secure under the static adversary model.*

Chapter 5

CRT-based Threshold Extensions

Secret and function sharing schemes can be enhanced by using various extensions. In this chapter, we will propose a CRT-based verifiable secret sharing (VSS) scheme and a proactive secret sharing (PSS) scheme. Also we will propose a CRT-based robust function sharing scheme. To the best of our knowledge the VSS and PSS schemes we propose are the first secure CRT-based SSSs. Besides, the robustness extension designed for the threshold RSA scheme described in Chapter 3 is the first one of its kind.

5.1 Verifiability

As described in Chapter 1, we call a SSS *verifiable* if each user can verify the correctness of his share in the dealer phase and no user can lie about his share in the combiner phase. Hence, neither the dealer nor the users can cheat in a VSS scheme. Verifiable secret sharing schemes based on Shamir's SSS have been proposed in the literature [27, 59]. These schemes have been extensively studied and used in threshold cryptography and secure multi-party computation [34, 58, 59].

There have been just two CRT-based VSS schemes by Iftene [40] and

Qiong et al. [62]. In this section, we show that these schemes are vulnerable to attacks where a corrupted dealer can distribute *inconsistent* shares without detection such that different coalitions will obtain different values for the secret.

A typical application of a VSS scheme is the joint random secret sharing (JRSS) primitive frequently used in threshold cryptography [34, 41, 58, 59]. In a JRSS scheme, all players act as a dealer and jointly generate and share a random secret. A simple CRT based JRSS scheme was described in Fig. 4.2 which does not use the verifiability feature.

In this section, we first show why existing attempts for a CRT-based verifiable secret sharing scheme fail by attacks on the existing schemes. We then propose a VSS scheme based on the Asmuth-Bloom secret sharing [2] and using this VSS scheme, we propose a JRSS scheme.

5.1.1 Analysis of Existing CRT-based VSS Schemes

There have been two different approaches to achieve VSS by a CRT-based secret sharing scheme. The first one, proposed by Iftene [40], obtains a VSS scheme from Mignotte's SSS [53] which is another CRT-based SSS similar to Asmuth-Bloom. Here, we adapt Iftene's approach to the Asmuth-Bloom SSS. The scheme is given in Figure 5.1.

If the dealer is honest and the discrete logarithm problem is hard, the scheme in Figure 5.1 is secure against a dishonest user because the verification data, ${g_i}^y \bmod p_i$, can be used to detect an invalid share from a corrupted user in the first step of the combiner phase.

However, if the dealer is dishonest, he can mount an attack despite the additional verification data above: Let y be an integer and $y_i = y \bmod m_i$ for $1 \leq i \leq n$. In the combiner phase of Asmuth-Bloom SSS, the minimum number of users required to obtain the secret is t; hence, $y = d + Am_0$ must be smaller than $M = \prod_{i=1}^{t} m_i$. Note that, to reconstruct the secret d, each coalition S must first compute $y \bmod M_S$ where $M_S \geq M$. If the dealer distributes the shares

- *Dealer Phase:* To share a secret $d \in \mathbb{Z}_{m_0}$ among a group of n users with verifiable shares, the dealer does the following:

  1. Use the dealing procedure of the Asmuth-Bloom SSS to obtain the shares $y_i = y \bmod m_i$ for each $1 \leq i \leq n$ where $y = d + Am_0 < M$. Choose m_is such that each $p_i = 2m_i + 1$ is also a prime.
  2. Let $g_i \in \mathbb{Z}_{p_i}^*$ be an element of order m_i. The dealer sends y_i to the ith user privately and makes the values p_i, g_i and $z_i = g_i^y \bmod p_i$ public for $1 \leq i \leq n$. The ith user can find whether his share is valid or not by checking
  $$z_i \stackrel{?}{=} {g_i}^{y_i} \bmod p_i. \tag{5.1}$$

- *Combiner Phase:* Let S be a coalition gathered to construct the secret.

  1. The share y_i of user $i \in S$ can be verified by the other users in S by the verification equation $z_i \stackrel{?}{=} {g_i}^{y_i} \bmod p_i$.
  2. If all shares are valid then the coalition S can obtain the secret d: First, the ith user computes
  $$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S.$$
  3. Then the users compute
  $$y = \left(\sum_{i \in S} u_i \right) \bmod M_S$$
  and obtain the secret d by computing $d = y \bmod m_0$.

Figure 5.1: Iftene's CRT-based VSS extension.

for some $y > M$, then y will be greater than M_S for some coalition S of size t. Hence, S may not compute the correct y value and the correct secret d even though $y_i = y \bmod m_i$ for all i. Therefore, the given VSS scheme cannot detect this kind of inconsistent shares from the dealer where different coalitions end up with different d values. The same problem also arises in Iftene's original VSS scheme [40].

Another VSS scheme based on Asmuth-Bloom secret sharing was proposed by Qiong et al. [62]. Their approach is similar to the VSS of Pedersen [59] based on Shamir's SSS. Their scheme is given in Figure 5.2.

- *Dealer Phase:* To share a secret $d \in \mathbb{Z}_{m_0}$ among a group of n users with verifiable shares, the dealer does the following:

  1. Use the dealing procedure of the Asmuth-Bloom SSS to obtain the shares $y_i = y \bmod m_i$ for all $1 \leq i \leq n$ where $y = d + Am_0 < M$.
  2. Let p, q be primes such that $q|(p-1)$. Construct the unique polynomial $f(x) \in \mathbb{Z}_q[x]$ where $deg(f(x)) = n-1$ and $f(m_i) = y_i$. Construct a random polynomial $f'(x) \in \mathbb{Z}_q[x]$ where $deg(f'(x)) = n-1$. Let $z_i = f'(m_i)$ for all $1 \leq i \leq n$.
  3. Let $g \in \mathbb{Z}_p$ with order q, h be a random integer in the group generated by g and $E(a,b) = g^a h^b \bmod p$ for inputs $a, b \in \mathbb{Z}_q^*$. Compute
  $$E_i = E(f_i, f_i') = g^{f_i} h^{f_i'} \bmod p,$$
  where f_i and f_i' are the $(i-1)$th coefficients of $f(x)$ and $f'(x)$, respectively, for all $1 \leq i \leq n$. Broadcast E_is to all users.
  4. Send (y_i, z_i) secretly to the ith user for all $1 \leq i \leq n$.
  5. Each user checks
  $$\begin{aligned} E(y_i, z_i) \stackrel{?}{\equiv} \prod_{j=1}^{n} {E_j}^{{m_i}^{j-1}} &\equiv \prod_{j=1}^{n} g^{f_j {m_i}^{j-1}} \prod_{j=1}^{n} h^{f_j' {m_i}^{j-1}} \\ &\equiv g^{y_i} h^{z_i} \pmod{p} \end{aligned} \tag{5.2}$$
  to verify the validity of his share.

- *Combiner Phase:* Let S be a coalition gathered to construct the secret.

  1. The share (y_i, z_i) of user $i \in S$ can be verified by the other users in S with the verification equality $E(y_i, z_i) \stackrel{?}{\equiv} \prod_{j=1}^{n} {E_j}^{{m_i}^{j-1}} \pmod{p}$.
  2. If all shares are valid; the coalition S can obtain the secret d by using the reconstruction procedure described in Section 2.1.

Figure 5.2: Qiong et al.'s CRT-based VSS extension.

As the scheme shows, Qiong et al. treated the shares of Asmuth-Bloom SSS as points on a degree-$(n-1)$ polynomial and adopted the approach of Pedersen by evaluating the polynomial in the exponent to verify the shares. If the dealer is honest, the scheme in Figure 5.2 is secure because the verification data can be used to detect an invalid share from a corrupted user in the first step of the combiner phase.

However, similar to the attack on Iftene's VSS scheme, if the dealer uses some $y > M$ and computes the verification data by using the shares $y_i = y \bmod m_i$, $1 \leq i \leq n$, the verification equation (5.2) holds for each user. But, for a coalition S where $y > M_S$, the coalition S cannot compute the correct y value and the secret d.

Note that Iftene's VSS scheme uses a separate verification data for each user; hence even if all the verification equations hold, the secret can still be inconsistent for different coalitions. Quiong et al.'s VSS scheme generates a polynomial $f(x)$ from the shares as in Feldman's and Pedersen's VSS schemes. This polynomial is used to check all verification equations. But Asmuth-Bloom SSS scheme depends on the CRT and unlike Shamir's SSS, here f is not inherently related to the shares. Hence, even if all the equations hold, the shares can still be inconsistent as we have shown.

5.1.2 Verifiable Secret Sharing with Asmuth-Bloom SSS

As discussed in Section 5.1.1, existing CRT-based VSS schemes in the literature cannot prevent a dealer from cheating. To solve this problem, we will use a range proof technique originally proposed by Boudot [11] and modified by Cao et al. [14].

5.1.2.1 Range Proof Techniques

Boudot [11] proposed an efficient and non-interactive technique to prove that a committed number lies within an interval. He used the Fujisaki-Okamoto commitment scheme [31], where the commitment of a number y with bases (g, h) is

computed as

$$E = E(y, r) = g^y h^r \bmod N$$

where g is an element in $\mathbb{Z}_N^*$, h is an element of the group generated by g, and r is a random integer. As proved in [11, 31], this commitment scheme is statistically secure assuming the factorization of N is not known.

After Boudot, Cao et al. [14] applied the same proof technique with a different commitment scheme

$$E = E(y) = g^y \bmod N$$

to obtain shorter range proofs. Here, we will use Cao et al.'s non-interactive range-proof scheme as a black box. For further details, we refer the user to [11, 14]. For our needs, we modified the commitment scheme as

$$E = E(y) = g^y \bmod PN$$

where $P = \prod_{i=1}^{n} p_i$ and N is an RSA composite whose factorization is secret. Note that even if $\phi(P)$ is known, $\phi(PN)$ cannot be computed since $\phi(N)$ is secret. Throughout the section, we will use $\text{RngPrf}(E(y), M)$ to denote the range proof that a secret integer y committed with $E(y)$ is in the interval $[0, M)$.

5.1.2.2 A CRT-based VSS Scheme

In our VSS scheme, the RSA composite N is an integer generated jointly by the users and the dealer where its prime factorization is not known. Such an integer satisfying these constraints can be generated by using the protocols proposed for shared RSA key generation [10, 30] at the beginning of the protocol. Note that we do not need the private and the public RSA exponents in our VSS scheme as in the original protocols [10, 30]; hence those parts of the protocols can be omitted.

Quisquater et al. [63] showed that when m_is are chosen as consecutive primes, the scheme has better security properties. For CRT-based VSS, we will also assume that all m_is are prime and we will choose them such that $p_i = 2m_i + 1$ is also a prime for $1 \leq i \leq n$.

Let $g_i \in \mathbb{Z}_{p_i}^*$ be an element of order m_i. Let $P = \prod_{i=1}^{n} p_i$ and

$$g = \left(\sum_{i=1}^{n} g_i P_i' \frac{P}{p_i} \right) \bmod P \tag{5.3}$$

where $P_i' = \left(\frac{P}{p_i}\right)^{-1} \bmod p_i$ for all $1 \leq i \leq n$, i.e., g is the unique integer in $\mathbb{Z}_P$ satisfying $g_i = g \bmod p_i$ for all i. Our VSS scheme is described in Figure 5.3.

- *Dealer Phase:* To share a secret $d \in \mathbb{Z}_{m_0}$ among a group of n users with verifiable shares, the dealer does the following:
  1. Use the dealing procedure of the Asmuth-Bloom secret sharing scheme described in Section 2.1 to obtain the shares
     $$y_i = y \bmod m_i$$
     for each $1 \leq i \leq n$ where $y = d + Am_0 < M = \prod_{i=1}^{t} m_i$. Note that the m_is are large primes where $p_i = 2m_i + 1$ is also a prime for $1 \leq i \leq n$.
  2. Let N be an integer whose prime factorization is not known by the users and the dealer. Compute $E(y) = g^y \bmod PN$. Send y_i to the ith user secretly for all $1 \leq i \leq n$ and broadcast $(E(y), \mathrm{RngPrf}(E(y), M))$.
  3. The ith user checks
     $$g_i{}^{y_i} \stackrel{?}{\equiv} E(y) \pmod{p_i} \tag{5.4}$$
     to verify $y_i = y \bmod m_i$. Then he checks the validity of the range proof to verify $y < M$.
- *Combiner Phase:* Let S be a coalition gathered to construct the secret.
  1. The share y_i of user $i \in S$ can be verified by the other users in S with the verification equality $g_i{}^{y_i} \stackrel{?}{\equiv} E(y) \pmod{p_i}$.
  2. If all shares are valid, the participants can obtain the secret d by using the reconstruction procedure described in Section 2.1. Otherwise, the corrupted users are disqualified.

Figure 5.3: CRT-based verifiable secret sharing scheme.

5.1.2.3 Analysis of the Proposed VSS Scheme

We analyze the correctness of the scheme and its security against passive and active attackers below:

5.1.2.4 Correctness

Aside from the verification equation, the scheme uses the original Asmuth-Bloom scheme. Hence, for correctness, we only need to show that when the dealer and the users are honest, the verification equations in the dealer and combiner phases hold. Note that, the condition $y < M$ is checked in Step 3 of the dealer phase by using $\text{RngPrf}(E(y), M)$). Furthermore, for a valid share y_i,

$$\begin{aligned} E(y) \bmod p_i &= g^y \bmod PN \bmod p_i \\ &= g^y \bmod p_i \\ &= {g_i}^y \bmod p_i \\ &= {g_i}^{y_i} \bmod p_i. \end{aligned}$$

Hence if the dealer and the users behave honestly, the verification equation holds and the ith user verifies that his share is a residue modulo m_i of the integer $y < M$ committed with $E(y)$.

5.1.2.5 Security Analysis

For the security analysis, we will first show that the VSS is perfect, i.e., no coalition of size smaller than t can obtain any information about the secret.

Theorem 5.1.1. *For a passive adversary with $t-1$ shares in the VSS scheme, every candidate for the secret is equally likely, i.e., the probabilities* $\Pr(d = d')$ *and* $\Pr(d = d'')$ *are approximately equal for all $d', d'' \in \mathbb{Z}_{m_0}$.*

Proof. First we recall that the underlying SSS is perfect due to Theorem 2.2.1. Hence, given $t-1$ shares, all secret candidates are equally likely. Besides the shares, the only additional information a corrupted user can obtain is $E(y)$ and $\text{RngPrf}(E(y), M)$. Given that the discrete logarithm problem is hard and

Cao et al.'s range proof technique is computationally secure, the proposed VSS scheme is also computationally secure. □

The shares distributed by a dealer are said to be inconsistent if different coalitions of size at least t obtain different values for the secret. The following theorem proves that the dealer cannot distribute shares inconsistent with the secret.

Theorem 5.1.2. *A corrupted dealer cannot cheat in the VSS scheme without being detected. I.e., if the shares are inconsistent with the secret d then at least one verification equation does not hold.*

Proof. Let $U = \{1, \ldots, n\}$ be the set of all users. If the shares are inconsistent, for two coalitions S and S' with $|S|, |S'| \geq t$,

$$\left(\sum_{i \in S} y_i M'_{S,i} M_{S \setminus \{i\}}\right) \bmod M_S \neq \left(\sum_{i \in S'} y_i M'_{S',i} M_{S' \setminus \{i\}}\right) \bmod M_{S'}.$$

hence,

$$y = \left(\sum_{i=1}^{n} y_i M'_{U,i} M_{U \setminus \{i\}}\right) \bmod M_U > M.$$

If this is true then the dealer cannot provide a valid range proof $\mathrm{RngPrf}(E(y), M)$. So, when a user tries to verify that $y < M$, the range proof will not be verified.

If the dealer tries to use a different $y' \neq y$ value in the commitment $E(y')$ and generates a valid proof $\mathrm{RngPrf}(E(y'), M)$, the verification equation (5.4) will not hold for some user i.

Hence, the VSS scheme guarantees that the n distributed shares are consistent and they are residues of some number $y < M$. □

Theorem 5.1.3. *A user cannot cheat in the VSS scheme without being detected; i.e., if a share given in the combiner phase is inconsistent with the secret, then the verification equation does not hold.*

Proof. When a user i sends an incorrect share $y_i' \neq y_i = y \bmod m_i$ in the combiner phase, the verification equation

$$E(y) \stackrel{?}{\equiv} {g_i}^{y_i} \pmod{p_i}$$

will not hold because $E(y) = g^y \bmod PN$, $p_i|P$ and since the order of $g_i \in \mathbb{Z}_{p_i}$ is m_i, the only value that satisfies the verification equation is y_i. □

Therefore, we can say that the scheme is secure for up to $t-1$ corrupted users and no participant can cheat in any phase of the scheme.

5.1.3 Verifiable Joint Random Secret Sharing

As described above, JRSS protocols enable a group of users to jointly generate and share a secret where a trusted dealer is not available. Here we describe a JRSS scheme based on the VSS scheme described above. We first modify (2.2) used in the Asmuth-Bloom secret sharing scheme in Section 2.1 as

$$\prod_{i=1}^{t} m_i > n{m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{5.5}$$

We also change the definition of M as $M = \left\lfloor (\prod_{i=1}^{t} m_i)/n \right\rfloor$. The proposed JRSS scheme is given in Figure 5.4.

5.1.3.1 Analysis of the Proposed JRSS Scheme

5.1.3.2 Correctness

Observe that when all users behave honestly, the JRSS scheme works correctly. Let $y = \sum_{i \in \mathcal{B}} y^{(i)}$. It is easy to see that $y < \prod_{i=1}^{t} m_i$ since $y^{(i)} < M$ for all $i \in \mathcal{B}$, where $|\mathcal{B}| \leq n$ and $M = \left\lfloor (\prod_{i=1}^{t} m_i)/n \right\rfloor$. One can see that $y_j = y \bmod m_j$ for all

- *Dealing Phase:* To jointly share a secret $d \in \mathbb{Z}_{m_0}$ the users do the following:

  1. Each user chooses a secret $d_i \in \mathbb{Z}_{m_0}$ and shares it by using the VSS scheme as follows: He first computes

  $$y^{(i)} = d_i + A_i m_0$$

  where $y^{(i)} < M = \left\lfloor (\prod_{i=1}^{t} m_i)/n \right\rfloor$. Then the secret for the jth user is computed as

  $$y_j^{(i)} = y^{(i)} \bmod m_j.$$

  He sends $y_j^{(i)}$ to user j secretly for all $1 \leq i \leq n$ and broadcasts $(E(y^{(i)}), \mathrm{RngPrf}(E(y^{(i)}), M))$.

  2. After receiving shares the jth user verifies them by using the verification procedure in (5.4). Let $\mathcal{B}$ be the set of users whose shares are verified correctly. The jth user computes his overall share

  $$y_j = \left(\sum_{i \in \mathcal{B}} y_j^{(i)} \right) \bmod m_j$$

  by using the verified shares.

- *Combiner Phase:* Let S be a coalition of t users gathered to construct the secret.

  1. The share y_i of user $i \in S$ can be verified by the other users in S with the verification equation,

  $$g^{y_i} \stackrel{?}{\equiv} \left(\prod_{j \in \mathcal{B}} E(y^{(j)}) \right) \pmod{p_i}. \tag{5.6}$$

  2. If all shares are valid, the participants obtain the secret

  $$d = \left(\sum_{i \in \mathcal{B}} d_i \right) \bmod m_0$$

  by using the reconstruction procedure described in Section 2.1.

Figure 5.4: CRT-based verifiable joint random secret sharing scheme.

$j \in \mathcal{B}$ by checking

$$\begin{aligned} y \bmod m_j &= \left(\sum_{i \in \mathcal{B}} y^{(i)} \right) \bmod m_j \\ &= \left(\sum_{i \in \mathcal{B}} y_j^{(i)} \right) \bmod m_j \\ &= y_j \bmod m_j = y_j. \end{aligned}$$

Hence, each y_i satisfies $y_i = y \bmod m_i$ and $y < \prod_{i=1}^{t} m_i$; so, y can be constructed with t shares.

For correctness of the verification procedure in (5.6), one can observe that

$$\begin{aligned} \left(\prod_{i \in \mathcal{B}} E(y^{(i)}) \right) \bmod p_i &= g^{\sum_{i \in \mathcal{B}} y^{(i)}} \bmod p_i \\ &= g^y \bmod p_i = {g_i}^y \bmod p_i \\ &= {g_i}^{y_i} \bmod p_i. \end{aligned}$$

Hence, when every user behaves honestly, the proposed JRSS scheme works correctly.

5.1.3.3 Security

We will show that no coalition of size smaller than t can obtain any information about the secret.

Theorem 5.1.4. *For a passive adversary with $t-1$ shares in the JRSS scheme, every candidate for the secret is equally likely. I.e., the probabilities $\Pr(d = d')$ and $\Pr(d = d'')$ are approximately equal for all $d', d'' \in \mathbb{Z}_{m_0}$.*

Proof. Suppose the adversary corrupts $t-1$ users and just observes the inputs and outputs of the corrupted users without controlling their actions, i.e., the adversary is honest in user actions but curious about the secret. Let S' be the coalition of the users corrupted by the adversary. The shares are obtained when each user shares his partial secret d_i, i.e., the adversary will obtain $t-1$ share for

each d_i. We will prove that the probabilities that $d_i = d_i'$ and $d = d_i''$ are almost equal for two secret candidates $d_i', d_i'' \in \mathbb{Z}_{m_0}$.

We already proved that the Asmuth-Bloom SSS described in Section 2.1 is perfect with equation (2.2). By using the shares of S', the adversary can compute $y'^{(i)} = y^{(i)} \bmod M_{S'}$. But even with these shares, there are $\frac{M}{M_{S'}}$ consistent $y^{(i)}$s which are smaller than M and congruent to $y'^{(i)}$ modulo $M_{S'}$. By replacing (2.2) with (5.5) and changing the definition of M to $\left\lfloor (\prod_{i=1}^{t} m_i)/n \right\rfloor$, the value of the ratio

$$\frac{M}{M_{S'}} > \frac{M}{\prod_{i=1}^{t-1} m_{n-i+1}} \approx \frac{\prod_{i=1}^{t} m_i}{n \prod_{i=1}^{t-1} m_{n-i+1}}$$

is greater than ${m_0}^2$. Hence, even with $t-1$ shares, there are still ${m_0}^2$ candidates for each $y^{(i)}$ which is used to share the secret d_i. Since $\gcd(m_0, M_{S'}) = 1$, there are approximately m_0 $y^{(i)}$s, consistent with a secret candidate d_i'. Hence, for a secret candidate d_i' the probability that $d_i = d_i'$ is approximately equal to $\frac{1}{m_0}$ and the perfectness of the scheme is preserved.

Besides the shares, the only other information the adversary can observe is the commitments and range proofs. Given that the discrete logarithm problem is hard and Cao et al.'s range proof scheme is secure, the proposed JRSS scheme is also computationally secure. □

A corrupted user cannot cheat in the JRSS scheme without being detected. Since we are using a VSS scheme, while user i is sharing his partial secret d_i, the conditions of the Asmuth-Bloom SSS must be satisfied as proved in Theorem 5.1.2. Furthermore, if user i sends an incorrect share in the combiner phase, the verification equation (5.6) will not hold. As a result, we can say that the JRSS scheme is secure for up to $t-1$ corrupted users and no user can cheat in any phase of the scheme.

5.2 Proactivity

Another important extension in threshold cryptography is the *proactivity* feature of the secret sharing schemes. With this feature, a SSS has the capability of renewing the shares of the users without changing the long term secret such that any shares obtained by a corrupted party becomes obsolete. So far, no CRT-based proactive secret sharing (PSS) schemes have been proposed in the literature. By combining and extending the ideas used in the VSS and JRSS schemes described in Section 5.1, we propose a PSS scheme in this section.

Proactive SSS protocols enable the shareholders to jointly renew their shares without changing and revealing the long-term secret. By this feature, the shares compromised by an adversary can be made obsolete with the update process. Proactive secret sharing schemes are investigated by several researchers and various schemes have been proposed in the literature [19, 39, 55].

In a proactive SSS, at the end of a certain time period τ, first the corrupted users are identified in the detection procedure and then all such users are rebooted, i.e., the adversary is removed from the computers of the users and all of the past information is erased. Subsequently, the new shares of the rebooted users are recovered in the recovery procedure. Then, the shares of the remaining users are refreshed by a renewal procedure. At the end of this protocol, the long-term secret remains the same although the shares of the users for the next period are renewed. This update phase is repeated periodically at the end of each time period.

Adversary model: We assume the *mobile adversary model* of Herzberg et al. [39]. In this model, the adversary is allowed to move among players and can corrupt users at any time. The only restriction on the adversary is that he cannot corrupt more than $t-1$ distinct users in a time period where $t < n/2$ is the threshold of the secret sharing scheme and n is the number of users. If a user is corrupted during the course of the update phase executed at the end of time period τ, he is considered corrupted for both time periods τ and $\tau+1$. With this model, Herzberg at al. proposed an efficient and secure Shamir based proactive

SSS.

We use $\mathcal{A}_\tau$ to denote the set of users where an adversarial behavior is detected in their actions in time period τ and $\mathcal{B}_\tau$ to denote the set of remaining users. A user is disqualified and becomes a member of $\mathcal{A}_\tau$ if his share is inconsistent with the secret or if he tries to cheat in the share update phase. Each disqualified user is rebooted at the beginning of the update phase, i.e., all the information including the secret share is erased, hence the new share of a corrupted user must be recovered by the users in $\mathcal{B}_\tau$.

5.2.1 CRT-based Proactive Secret Sharing Scheme

To obtain a proactive SSS, we first modify the equation (2.1) used in the Asmuth-Bloom secret sharing scheme in Section 2.1 as

$$\prod_{i=1}^{t} m_i > n{m_0}^3 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{5.7}$$

We also change the definition of M as

$$M = \left\lfloor \frac{\prod_{i=1}^{t} m_i}{n m_0} \right\rfloor.$$

In the proposed proactive sharing scheme, first a secret is shared by a dealer as described in Figure 5.5 by using the VSS scheme proposed in Section 5.1.2.

Dealer Phase: The dealer shares a secret $d \in \mathbb{Z}_{m_0}$ by equation (5.7) and M, using the VSS scheme proposed in Section 5.1.2. Similar to the VSS, let $p_i = 2m_i+1$ be a prime for $0 \leq i \leq n$. As in the VSS, $y = d + Am_0$ is an integer smaller than M, $y_i = y \bmod m_i$ is the share of user i, and the commitment $E(y) = g^y \bmod PN$ is broadcast with the range proof for y.

Figure 5.5: CRT-based proactive SSS: The dealer phase.

The share update phase executed at the end of a time period τ has three phases:

1. *Detection*: If a user j is detected as corrupted he is rebooted and becomes a member of $\mathcal{A}_\tau$.

2. *Share Recovery*: The share of each rebooted user $j \in \mathcal{A}_\tau$ is reconstructed by the remaining users in $\mathcal{B}_\tau$.

3. *Share Renewal*: The users jointly share 0 by setting $d_i = 0$ for $1 \leq i \leq n$ in the JRSS protocol of Figure 5.4. Then they add these renewal shares to their previous ones and obtain their new shares.

5.2.1.1 Detection

If a user does not participate in a protocol where he is an active member, or if the information he sends does not verify correctly we say that an adversarial action is detected. However, when an adversary *silently* corrupts some users by only modifying their local data, we cannot detect such inconsistencies after the adversary detaches himself from the user. Hence, to protect proactiveness, we need to periodically test the correctness of users' local data. To do this, we will use the $E(y)$ values each player holds as in Figure 5.6.

Note that $t < n/2$ hence there are at least t honest users who had not been silently or actively corrupted by an adversary. Since the views of all honest users are the same, an inconsistent value will be detected by at least t users.

5.2.1.2 Share Recovery

At the beginning of the update phase, the shares of the rebooted users will still be missing. To recover the share of a rebooted user $j \in \mathcal{A}_{\tau-1}$, each user $\mathcal{B}_{\tau-1}$ shares a random multiple of $m_j \in [0, M)$; hence the sum of these shared values, which will be denoted by z, will be a multiple of m_j. This ensures that when the users in $\mathcal{B}_{\tau-1}$ add their old shares for y by the new ones, they obtain a share for an integer $y' = y + z$ where $y_j \equiv y \equiv y' \pmod{m_j}$. After obtaining a share for y', each user in $\mathcal{B}_{\tau-1}$ sends it to the jth user via a private channel so the jth

1. Let $E^{(i)}(y)$ denotes the local copy of the commitment $E(y)$ for user i. For detection, the ith user first checks the validity of his share by using the equation
$$y_i \stackrel{?}{=} E^{(i)}(y) \bmod p_i.$$
If the equation is not satisfied he broadcasts an inconsistency warning, reboots himself and becomes a member of $\mathcal{A}_\tau$.

2. If the equation is satisfied, after receiving each $E^{(i)}(y)$ for $i \in \mathcal{B}_\tau$, the jth user tries to find an inconsistency
$$E^{(i_1)}(y) \neq E^{(i_2)}(y)$$
for $i_1, i_2 \in \mathcal{B}_\tau$ and he broadcast an accusation either for user i_1, i_2 or both, depending which is inconsistent with at least t $E^{(i)}(y)$s.

3. If at least t accusations are broadcasted for a user i, he is rebooted with the users who did not accuse him.

Figure 5.6: CRT-based proactive SSS: The detection procedure.

user can compute y' and hence y_j. This share recovery procedure is described in detail in Figure 5.7.

5.2.1.3 Share Renewal

After the recovery procedure, each user $i \in \mathcal{B}_\tau$ has a share $y_i = y \bmod m_i$ where $y = d + Am_0 < M$. The idea used in this phase is similar to the one in the JRSS scheme described in Section 5.1.3. Instead of a random secret, each user shares 0 by some $y^{(i)} \equiv 0 \pmod{m_0}$, $y^{(i)} \in [0, M)$, hence the overall shared value will be a multiple of m_0. So, when a user adds his renewal shares with his old share y_i, he obtains a new share y_i'', which is a residue of an integer $y'' \in \left[0, \prod_{i=1}^{t} m_i\right)$ such that $d = y'' \bmod m_0$. In the next time period, y'' will be the new y. The share renewal procedure is described in Figure 5.8.

Note that y will remain less than $\prod_{i=1}^{t} m_i$ provided that m_0, which is a very large integer, is greater than the number of times the update procedure is applied,

1. To recover the share of a compromised user $j \in \mathcal{A}_\tau$ each user $i \in \mathcal{B}_\tau$ chooses an integer
$$y^{(i,j)} = A_i m_j$$
where A_i is a random integer such that $y^{(i,j)} < M$ and then shares it among $\mathcal{B}_\tau$ by computing the secrets
$$y_k^{(i,j)} = y^{(i,j)} \bmod m_k$$
for each user $k \in \mathcal{B}_\tau$. He sends $y_k^{(i,j)}$ to user k secretly and broadcasts $(E(y^{(i,j)}), \mathrm{RngPrf}(E(y^{(i,j)}), M))$.

2. After receiving the shares $y_k^{(i,j)}$ from each $i \in \mathcal{B}_\tau$, the kth player verifies them by using the verification procedure in equation (5.4). Also each commitment is checked by $E(y^{(i,j)}) \bmod p_j \stackrel{?}{=} 1$. If a verification equation does not hold for a user, he is disqualified.

3. The kth user computes his ephemeral secret
$$y'_k = \left(y_k + \sum_{i \in \mathcal{B}_\tau} y_k^{(i,j)} \right) \bmod m_k$$
and sends it to user j secretly.

4. After receiving the shares, y'_ks, from each user $k \in \mathcal{B}_\tau$, the jth player verifies them by using the verification procedure in equation (5.4) for y'. The verification data for $y' = y + \sum_{i \in \mathcal{B}_\tau} y^{(i,j)}$ can be computed as
$$E(y') = E(y) \prod_{i \in \mathcal{B}_\tau} E(y^{(i,j)}).$$
If a verification equation does not hold for a user, he is disqualified for time period τ and $\tau + 1$.

5. The compromised user jth computes
$$y' = \sum_{k \in \mathcal{B}_\tau} y'_k M'_{\mathcal{B}_\tau,k} M_{\mathcal{B}_\tau \setminus \{k\}} \bmod M_{\mathcal{B}_\tau}$$
where $M_{\mathcal{B}_\tau \setminus \{k\}} M'_{\mathcal{B}_\tau,k} \equiv 1 \pmod{m_k}$. He computes his share as $y_j = y' \bmod m_j$.

Figure 5.7: CRT-based proactive SSS: The share recovery procedure.

1. Each user i in $\mathcal{B}_\tau$ shares 0 by first computing

$$y^{(i)} = A_i m_0$$

where A_i is a random integer such that $y^{(i)} < M$. Then he computes the share for user $j \in \mathcal{B}_\tau$ as

$$y_j^{(i)} = y^{(i)} \bmod m_j.$$

He sends $y_j^{(i)}$ to each user j secretly and broadcasts $\big(E(y^{(i)}), \text{RngPrf}(E(y^{(i)}), M)\big)$.

2. After receiving the shares $y_j^{(i)}$ for $i \in \mathcal{B}_\tau$, the jth user verifies them by using the verification procedure in equation (5.4). Besides, each commitment is checked for $E(y^{(i)}) \stackrel{?}{\equiv} 1 \pmod{p_0}$. If the verification equation of user i does not hold, he is disqualified, i.e., he is moved from $\mathcal{B}_\tau$ to $\mathcal{A}_\tau$ and $\mathcal{A}_{\tau+1}$. The jth user updates his overall share as

$$y_j'' = \left(y_j + \sum_{i \in \mathcal{B}_\tau} y_j^{(i)} \right) \bmod m_j.$$

3. The new verification data for $y'' = y + \sum_{i \in \mathcal{B}_\tau} y^{(i)}$ is computed as

$$E(y'') = E(y) \prod_{i \in \mathcal{B}_\tau} E(y^{(i)}).$$

Figure 5.8: CRT-based proactive SSS: The share renewal procedure.

since $y^{(i)} < M$, $|\mathcal{B}_\tau| < n$, and

$$m_0 \sum_{i \in \mathcal{B}_\tau} y^{(i)} < nm_0 M = nm_0 \left\lfloor \frac{\prod_{i=1}^{t} m_i}{nm_0} \right\rfloor \leq \prod_{i=1}^{t} m_i.$$

5.2.2 Security Analysis

Assume that the update phase is started at the end of the τth time period. In the share recovery procedure, the participants share z which is equivalent to 0 modulo m_j for a rebooted user j. Also, in the share renewal procedure, the participants jointly share $y'' - y$, where y'' is the new shared integer and y is the previous one. With the following theorems, we will prove that the perfectness condition is preserved in the dealing phase and the shared integers in the next two phases are not computable by a passive adversary.

Theorem 5.2.1. *The modified secret sharing scheme with the new*

$$M = \left\lfloor \frac{\prod_{i=1}^{t} m_i}{nm_0} \right\rfloor$$

and equation (5.7) is perfect in the sense that the probabilities $\Pr(d = d')$ *and* $\Pr(d = d'')$ *are approximately equal for all* $d', d'' \in \mathbb{Z}_{m_0}$.

Proof. Let S' be a corrupted coalition of $t-1$ users. For perfectness, we need to check the value of $M/M_{S'}$ which is

$$\frac{M}{M_{S'}} > \frac{\prod_{i=1}^{t} m_i}{nm_0 \prod_{i=n-t+1}^{n} m_i} > {m_0}^2$$

due to equation (5.7). Similar to the proof of Theorem 2.2.1, we can say that the perfectness condition is preserved. □

Lemma 5.2.2. *For a passive adversary that has corrupted* $t-2$ *users in the recovery procedure, there are at least* ${m_0}^2$ *possible candidates for each* $y^{(i,j)}$ *used.*

Proof. Let S' be the set of $t-2$ corrupted users. In the recovery procedure, first each user i shares a $y^{(i,j)}$ where adversary has $t-2$ shares for each of them, i.e.,

$y_k^{(i,j)}$ for $k \in S' \setminus \{j\}$. So, for a shared value $y^{(i,j)}$, the adversary can only have the shares of S' and an additional information that $y^{(i,j)} \equiv 0 \pmod{m_j}$. Hence, although the adversary can obtain $y^{(i,j)} \bmod M_{S'}$, there are still

$$\frac{M}{M_{S'}} = \frac{\prod_{i=1}^{t} m_i}{n m_0 M_{S'}} > {m_0}^2$$

candidates for $y^{(i,j)}$ since $y^{(i,j)} < M$. □

Lemma 5.2.3. *Let j be the rebooted user whose share is being recovered in the recovery procedure. For a passive adversary that has corrupted $t-1$ users including j, there are at least ${m_0}^2$ possible values for each uncompromised y_i, the secret share of user i.*

Proof. In Step 3 of the recovery procedure described in Figure 5.7, an honest user i computes his ephemeral secret

$$y_i' = \left(y_i + \sum_{k \in \mathcal{B}_\tau} y_i^{(k,j)} \right) \bmod m_i$$

and sends it to user j who has been corrupted by the adversary. Note that y_i is masked with $y_i^{(k,j)}$s where

$$y_i^{(k,j)} = y^{(k,j)} \bmod m_i,$$

and due to Lemma 5.2.2, from the adversary's point of view there are at least ${m_0}^2$ candidates, with the same remainder in modulo $M_{S'}$, for each $y^{(k,j)}$. Hence there are at least ${m_0}^2$ candidates for each $y_i^{(k,j)}$ since $(m_i, M_{S'}) = 1$. This also proves that there are at least ${m_0}^2$ candidates for $y_i = d + Am_0 \bmod m_i$. □

Theorem 5.2.4. *For a passive adversary in the recovery procedure, two secrets $d', d'' \in \mathbb{Z}_{m_0}$ are equally likely.*

Proof. Let j be the rebooted user whose share is being recovered. Since user j was corrupted in time period τ, the adversary can have at most $t-2$ additional users corrupted in the recovery procedure. Beside these $t-2$ users, the adversary is allowed to corrupt only the jth user again. Due to the mobile adversary model

this is the best the adversary can do. Let S' be the set of $t-1$ corrupted users including user j.

From Lemma 5.2.3, we know that there are at least ${m_0}^2$ candidates for $y_i = d + Am_0 \bmod m_i$. Since $\gcd(m_0, m_i) = 1$ these ${m_0}^2$ candidates covers all m_0 secret candidates at least m_0 times. Hence, all secret candidates are equally likely. □

Lemma 5.2.5. *For a passive adversary that has corrupted $t-1$ users in the share renewal procedure, there are at least m_0 possible candidates for each $y^{(i)}$.*

Proof. Assume that the adversary corrupted $t-1$ users in time period τ without being detected. Let S' denote this set of corrupted users. Considering $M \approx (\prod_{i=1}^{t} m_i)/(nm_0)$ we know that

$$M = \frac{\prod_{i=1}^{t} m_i}{nm_0} > {m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1} > {m_0}^2 M_{S'}$$

due to equation (5.7).

For a shared value $y^{(i)} = A_i m_0$, the adversary will know that $y^{(i)} \equiv 0 \pmod{m_0}$. Since $y^{(i)} < M$, there are $\frac{M}{m_0} > m_0 M_{S'}$ candidates for $y^{(i)} \equiv 0 \pmod{m_0}$. Besides, the adversary can compute $y^{(i)} \bmod M_{S'}$ by using the $t-1$ shares he obtained for $y^{(i)}$. But, there are still $\frac{M}{m_0 M_{S'}} > m_0$ candidates for $y^{(i)}$. □

Theorem 5.2.6. *An adversary with $t-1$ corrupted shares in the share renewal procedure cannot compute a new share in time period $\tau+1$ from an old share he has from time period τ.*

Proof. In Step 2 of the renewal procedure describe in Figure 5.8, the jth user updates his overall share as

$$y_j'' = \left(y_j + \sum_{i \in \mathcal{B}_\tau} y_j^{(i)} \right) \bmod m_j.$$

From Lemma 5.2.5, there is at least m_0 possible candidates for $y^{(i)}$ and $\gcd(m_j, m_0 M_{S'}) = 1$. Hence, there are at least m_0 possible candidates for each

$$y_j^{(i)} = y^{(i)} \bmod m_j.$$

Hence, the adversary cannot compute y_j'' even if he knows the old share y_j. □

By Theorems 5.2.4 and 5.2.6, the proposed PSS scheme is secure against passive adversaries in Herzberg et al.'s mobile adversary model.

5.2.2.1 Security Analysis for an Active Adversary

As proved in Sections 5.1.2.5 and 5.1.3.3, in the proposed VSS and JRSS schemes, if a user tries to cheat by sending inconsistent information he will be detected easily since some verification equations will not hold.

In the share renewal and share recovery phases, we use modified versions of the JRSS scheme where the shared values are congruent to 0 with respect to moduli m_0 and m_j, respectively, where user j has been rebooted before the execution of the update phase. To verify these restrictions, in the 2nd step of Figure 5.8, a user j verifies his share for $y^{(i)}$ by using the verification procedure in equation (5.4) and checks

$$E(y^{(i)}) \stackrel{?}{\equiv} 1 \pmod{p_0}.$$

Also in the 2nd step of Figure 5.7, a user k verifies his share for $y^{(i,j)}$ by using the verification procedure in equation 5.4 and checks

$$E(y^{(i,j)}) \stackrel{?}{\equiv} 1 \pmod{p_j}.$$

Note that the other restrictions are also verified since they are automatically checked by the proposed VSS scheme. Therefore, an active adversary cannot send an inconsistent data without being detected.

5.3 Robustness

We say that a function sharing scheme is *robust* if it can withstand participation of corrupt users in the function evaluation phase. In a robust FSS, a detection mechanism is used to identify the corrupted partial results so that, the corrupted users can be eliminated. The FSSs proposed in Chapters 3 and 4 do not have

the robustness property and, to the best of our knowledge, no CRT-based robust and secure function sharing scheme exists in the literature.

In this section, we investigate how CRT-based threshold schemes can be enhanced with the robustness property. We first give a robust threshold function sharing scheme for the RSA cryptosystem. Then we apply the ideas to the ElGamal and Paillier decryption functions. For RSA and Paillier, we use the threshold schemes proposed in Chapter 3. For ElGamal, we work with a modified version of the ElGamal decryption scheme by Wei et al. [72]. All of the proposed schemes are provably secure against a static adversary under the random oracle model [5].

In achieving robustness, we make use of a non-interactive protocol designed to prove equality of discrete logarithms [11, 15, 68]. The original interactive protocol was proposed by Chaum et al [15] and improved by Chaum and Pedersen [16]. Later, Shoup [68] and, Boudot and Traoré [12] developed a non-interactive version of the protocol.

In the original Asmuth-Bloom scheme, m_0 is not needed until the last step of the combiner phase but still it is a public value. To avoid confusion, we emphasize that it will be secret for the robust FSSs proposed in this section.

5.3.1 Robust Sharing of the RSA Function

To enhance the threshold cryptosystems with the robustness property, we use a non-interactive protocol proposed to prove equality of two discrete logarithms with respect to different moduli. The interactive protocol, which was originally proposed by Chaum et al [15] for the same moduli, was modified by Shoup and used to make a threshold RSA signature scheme robust [68]. He used Shamir's SSS as the underlying SSS to propose a practical and robust threshold RSA signature scheme. In Shamir's SSS, the secret is reconstructed by using Lagrange's polynomial evaluation formula and all participants use the same modulus which does not depend on the coalition. On the other hand, in the direct solution used in the the CRT-based threshold RSA scheme described in Section 3.1, the

definition of

$$u_i = \left(y_i M'_{S,i} \bmod m_i\right) M_{S\setminus\{i\}}$$

shows that we need different moduli for each user. For robustness, we need to check the correctness of u_i for each user i in the function evaluation phase. We modified the protocol in [68] for the case of different moduli as Boudot and Traoré [11] did to obtain efficient publicly verifiable secret sharing schemes.

To obtain robustness, we first modify the dealer phase of the Asmuth-Bloom SSS and add the constraint that

$$p_i = 2m_i + 1$$

be a prime for each $1 \leq i \leq n$. These values will be the moduli used to construct/verify the *proof of correctness* for each user. The robustness extension described below can be used to make the CRT-based threshold RSA signature scheme in Section 3.1 robust. We only give the additions for the robustness extension here since the other phases are the same.

- *Setup*: Use Asmuth-Bloom SSS for sharing d with $m_0 = \phi(N)$. Let g_i be an element of order m_i in $\mathbb{Z}^*_{p_i}$. Broadcast g_i and the public verification data

  $$v_i = {g_i}^{y_i} \bmod p_i$$

  for each user i, $1 \leq i \leq n$.

- *Generating the proof of correctness*: Let w be the hashed message to be signed and suppose the range of the hash function is $\mathbb{Z}^*_N$. Assume a coalition S of size t participated in the signing phase. Let $h : \{0,1\}^* \rightarrow \{0, \ldots, 2^{L_1} - 1\}$ be a hash function where L_1 is another security parameter. Let

  $$w' = w^{M_{S\setminus\{i\}}} \bmod N,$$

  $$v'_i = {v_i}^{M'_{S,i}} \bmod p_i,$$

  $$z_i = y_i M'_{S,i} \bmod m_i.$$

Each user $i \in S$ first computes

$$W = w'^{r} \bmod N,$$
$$G = {g_i}^{r} \bmod p_i$$

where $r \in_R \{0, \ldots, 2^{L(m_i)+2L_1}\}$. Then he computes the proof as

$$\sigma_i = h(w', g_i, s_i, v_i', W, G),$$

$$D_i = r + \sigma_i z_i \in \mathbb{Z}$$

and sends the proof (σ_i, D_i) along with the partial signature s_i.

- *Verifying the proof of correctness:* The proof (σ_i, D_i) for the ith user can be verified by checking

$$\sigma_i \stackrel{?}{=} h(w', g_i, s_i, v_i', {w'}^{D_i} {s_i}^{-\sigma_i} \bmod N, {g_i}^{D_i} {v_i'}^{-\sigma_i} \bmod p_i). \tag{5.8}$$

Note that the above scheme can also be used to obtain a robust threshold RSA decryption scheme. Since RSA signature and decryption functions are mostly identical, we omit the details.

5.3.1.1 Security Analysis

Here we will prove that the proposed threshold RSA signature scheme is secure (i.e. existentially non-forgeable against an adaptive chosen message attack), provided that the RSA problem is intractable (i.e. RSA function is a one-way trapdoor function [18]). We assume the same static adversary model of Chapter 3 where the adversary controls exactly $t-1$ users and chooses them at the beginning of the attack. In this model, the adversary obtains all secret information of the corrupted users and the public parameters of the cryptosystem. She can control the actions of the corrupted users, ask for partial signatures of the messages of her choice, but she cannot corrupt another user in the course of an attack, i.e., the adversary is static in that sense.

First we will analyze the proof of correctness. For generating and verifying the proof of correctness, the following properties holds:

- *Completeness:* If the ith user is honest then the proof succeeds since

$$w'^{D_i} s_i^{-\sigma_i} = w'^r \bmod N,$$
$$g_i^{D_i} v_i'^{-\sigma_i} = g_i^r \bmod p_i.$$

- *Soundness:* To prove the soundness, we will use a lemma by Poupard and Stern [61] which states that if the prover knows $(a, b, \sigma, \sigma', D, D')$ such that $a^D b^\sigma \equiv a^{D'} b^{\sigma'} \pmod K$ for an integer K, then he knows the discrete logarithm of b in base a unless he knows the factorization of K.

  Let us define $\Psi : \mathbb{Z}^*_{Np_i} \to \mathbb{Z}^*_N \times \mathbb{Z}^*_{p_i}$ be the CRT isomorphism, i.e., $x \to (x \bmod N, x \bmod p_i)$ for $x \in \mathbb{Z}^*_{Np_i}$. Note that $\gcd(N, p_i) = 1$. Let $g = \Psi^{-1}(w', g_i)$, $v = \Psi^{-1}(s_i, v_i')$ and $\tau = \Psi^{-1}(W, G)$. Given W and G, if the ith user can compute valid proofs (σ, D) and (σ', D') then we have

$$\tau = g^D v^\sigma \bmod Np_i = g^{D'} v^{\sigma'} \bmod Np_i$$

  and according to the lemma above, the ith user knows u_i unless he can completely factor Np_i. Since the factorization of N is secret we can say that if the proof is a valid proof then the discrete logarithms are equal in modm_i and the prover knows this discrete logarithm. Hence, an adversary cannot impersonate a user without knowing his share. Similar to Boudot and Traore [11], a range check on D_i might be necessary while verifying the proof of correctness to detect incorrect partial signatures from users with valid shares.

- *Zero-Knowledge Simulatability:* To prove the zero-knowledge simulatability, we will use the random oracle model for the hash function h and construct a simple simulator. When an uncorrupted user wants to create a proof (σ_i, D_i) for a message w and partial signature s_i, the simulator returns

$$\sigma_i \in_R \{0, \ldots, 2^{L_1} - 1\}$$

  and

$$D_i \in_R \{0, \ldots, 2^{L(m_i)+2L_1} - 1\}$$

and sets the value of the oracle at

$$(w', g_i, s_i, v_i', {w'}^{D_i} {s_i}^{-\sigma_i} \bmod N, {g_i}^{D_i} {v_i'}^{-\sigma_i} \bmod p_i)$$

as σ_i. Note that, the value of the random oracle is not defined at this point but with negligible probability. When a corrupted user queries the oracle, if the value of the oracle was already set the simulator returns that value otherwise it returns a random one. It is obvious that the distribution of the output of the simulator is statistically indistinguishable from the real output.

To reduce the security of the proposed threshold RSA signature scheme to the security of the standard RSA signature scheme, the following proof constructs another simulator.

Theorem 5.3.1. *Given that the standard RSA signature scheme is secure, the threshold RSA signature scheme is robust and secure under the static adversary model.*

Proof. To reduce the problem of breaking the standard RSA signature scheme to breaking the proposed threshold scheme, we will simulate the threshold protocol with no information on the secret where the output of the simulator is indistinguishable from the adversary's point of view. Afterwards, we will show that the secrecy of the private key d is not disrupted by the values obtained by the adversary. Thus, if the threshold RSA scheme is not secure, i.e., an adversary who controls $t-1$ users can forge signatures in the threshold scheme, one can use this simulator to forge a signature in the standard RSA scheme.

The simulator's actions are as same as the the one described in the proof of Theorem 3.1.1 which is used to prove the security of the underlying CRT-based threshold RSA scheme. In addition to that, the simulator computes the public verification data of the users in S' as $v_j = g^{y_j} \bmod p_j$ for $j \in S'$. For other users $i \notin S'$, the simulator chooses a random integer $y_i \in_R \mathbb{Z}_{m_i}$ and sets $v_i = g^{y_i} \bmod p_i$. Note that $\gcd(N, p_i) = 1$. So the public verification data

generated by the simulator are computationally indistinguishable from the real ones.

Consequently, the output of the simulator is indistinguishable from a real instance from the adversary's point of view, and hence the simulator can be used to forge a signature in the standard RSA scheme if the threshold RSA scheme can be broken. □

5.3.2 Robustness in Other CRT-based Threshold Schemes

The robustness extension given in Section 5.3.1 can be applied to other CRT-based threshold schemes as well. Here we describe how to adapt the extension to the CRT-based threshold Paillier and ElGamal function sharing schemes.

5.3.2.1 Robust Sharing of the Paillier Decryption Function

As described in Section 3.2.2, Paillier's probabilistic cryptosystem [57] is a member of a different class of cryptosystems where the message is used in the exponent of the encryption operation. The robustness extension can be applied to the Paillier cryptosystem as follows:

- *Setup*: Let $N = pq$ be the product of two large primes and $\lambda = \text{lcm}(p-1, q-1)$. Use Asmuth-Bloom SSS for sharing λ with $m_0 = \phi(N^2) = N\phi(N)$. Let $g_i \in \mathbb{Z}_{p_i}^*$ be an element with order m_i in $\mathbb{Z}_{p_i}^*$. Broadcast the public verification data g_i and
$$v_i = g_i^{y_i} \bmod p_i$$
for each user i, $1 \leq i \leq n$.

- *Generating the proof of correctness*: Let $h : \{0,1\}^* \rightarrow \{0, \ldots, 2^{L_1} - 1\}$ be a hash function where L_1 is another security parameter. Let
$$c' = c^{M_{S\setminus\{i\}}} \bmod N^2,$$

$$v'_i = {v_i}^{M'_{S,i}} \bmod p_i,$$

$$z_i = y_i M'_{S,i} \bmod m_i.$$

Each user $i \in S$ first computes

$$W = {c'}^r \bmod N^2,$$

$$G = {g_i}^r \bmod p_i$$

where $r \in_R \{0, \ldots, 2^{L(m_i)+2L_1}\}$. Then he computes the proof as

$$\sigma_i = h(c', g_i, s_i, v'_i, W, G),$$

$$D_i = r + \sigma_i z_i \in \mathbb{Z}$$

and sends the proof (σ_i, D_i) along with the partial decryption s_i.

- *Verifying the proof of correctness:* The proof (σ_i, D_i) for the ith user can be verified by checking

$$\sigma_i \stackrel{?}{=} h(c', g_i, s_i, v'_i, {c'}^{D_i} {s_i}^{-\sigma_i} \bmod N, {g_i}^{D_i} {v'_i}^{-\sigma_i} \bmod p_i). \tag{5.9}$$

If the ith user is honest then the proof succeeds since ${c'}^{D_i} {s_i}^{-\sigma_i} = {c'}^r \bmod N^2$ and ${g_i}^{D_i} {v'_i}^{-\sigma_i} = {g_i}^r \bmod p_i$. The soundness property can be proved with a proof similar to the proof of Theorem 1. Note that $\gcd(N^2, p_i) = 1$ for all users and $\phi(N^2) = N\phi(N)$ is secret. A similar proof can be given for the zero knowledge simulatability as the one in Section 5.3.1.1.

5.3.2.2 Robust Sharing of the ElGamal Decryption Function

Adapting our robustness extension to the threshold ElGamal scheme given in Chapter 3.1 is slightly more complicated than it is for the Paillier's cryptosystem, because $\phi(p) = p - 1$ is public. A simple solution for this problem is to extend the modulus to $N = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes. There exist versions of the ElGamal encryption scheme in the literature with a composite modulus instead of p. For example, Wei et al. [72] modified the standard ElGamal scheme to obtain a hidden-order ElGamal scheme. They proved that their scheme

is as secure as each of the standard RSA and ElGamal cryptosystems. Here, we give the description of a robust, CRT-based threshold scheme for Wei et al.'s version of the ElGamal encryption.

- *Setup*: In the ElGamal setup phase, choose $p = 2p' + 1$ and $q = 2q' + 1$ be large primes such that p' and q' are also prime numbers. Let $N = pq$ and let g_p and g_q be generators of $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, respectively. Choose $\alpha_p \in_R \mathbb{Z}_p^*$ and $\alpha_q \in_R \mathbb{Z}_q^*$ such that $\gcd(p-1, q-1) \mid (\alpha_p - \alpha_q)$. The secret key $\alpha \in \mathbb{Z}_{\lambda(N)}$ is the unique solution of the congruence system

$$\begin{aligned} \alpha &\equiv \alpha_p \pmod{p-1}, \\ \alpha &\equiv \alpha_q \pmod{q-1} \end{aligned}$$

  where $\lambda(N) = 2p'q'$ is the Carmichael number of N. Similarly, the public key $\beta \in \mathbb{Z}_N$ is the unique solution of congruence system

$$\begin{aligned} \beta &\equiv {g_p}^{\alpha_p} \pmod{p}, \\ \beta &\equiv {g_q}^{\alpha_q} \pmod{q}. \end{aligned}$$

  Let g be the unique solution of the congruence system

$$\begin{aligned} g &\equiv g_p \pmod{p}, \\ g &\equiv g_q \pmod{q} \end{aligned}$$

  and α and (β, g, N) be the private and the public keys, respectively. Note that $\beta = g^\alpha \bmod N$. Use Asmuth-Bloom SSS for sharing the private key α with $m_0 = 2p'q'$. Let $g_i \in \mathbb{Z}_{p_i}^*$ be an element with order m_i in $\mathbb{Z}_{p_i}^*$. Broadcast the public verification data g_i and $v_i = g_i^{y_i} \bmod p_i$ for each user i, $1 \leq i \leq n$.

- *Encryption*: Given a message $w \in \mathbb{Z}_N$, the ciphertext $c = (c_1, c_2)$ is computed as

$$\begin{aligned} c_1 &= g^k \bmod N, \\ c_2 &= \beta^k w \bmod N \end{aligned}$$

  where k is a random integer from $\{1, \ldots, N-1\}$.

- *Decryption*: Let (c_1, c_2) be the ciphertext to be decrypted where $c_1 = g^k \bmod N$ for some $k \in \{1, \ldots, N-1\}$ and $c_2 = \beta^k w \bmod N$ where w is the message. The coalition S of t users wants to obtain the message $w = sc_2 \bmod N$ for the *decryptor* $s = (c_1^{\alpha})^{-1} \bmod N$.

  - *Generating the partial results*: Each user $i \in S$ computes

$$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S, \tag{5.10}$$

$$s_i = {c_1}^{-u_i} \bmod N,$$

$$\beta_i = g^{u_i} \bmod N. \tag{5.11}$$

  - *Generating the proof of correctness*: Let $h : \{0,1\}^* \to \{0, \ldots, 2^{L_1} - 1\}$ be a hash function where L_1 is another security parameter. Let

$$c'_1 = {c_1}^{M_{S\setminus\{i\}}} \bmod N,$$

$$v'_i = {v_i}^{M'_{S,i}} \bmod p_i,$$

$$z_i = y_i M'_{S,i} \bmod m_i.$$

    Each user $i \in S$ first computes

$$W = {c'_1}^{r} \bmod N,$$

$$G = {g_i}^{r} \bmod p_i$$

    where $r \in_R \{0, \ldots, 2^{L(m_i)+2L_1}\}$. Then he computes the proof as

$$\sigma_i = h(c'_1, g_i, s_i, v'_i, W, G),$$

$$D_i = r + \sigma_i z_i \in \mathbb{Z}$$

    and sends the proof (σ_i, D_i) along with s_i.

  - *Verifying the proof of correctness:* The proof (σ_i, D_i) for the ith user can be verified by checking

$$\sigma_i \stackrel{?}{=} h(c'_1, g_i, s_i, v'_i, {c'_1}^{D_i} {s_i}^{-\sigma_i} \bmod N, {g_i}^{D_i} {v'_i}^{-\sigma_i} \bmod p_i).$$

– *Combining the partial results*: The incomplete decryptor $\overline{s}$ is obtained by combining the s_i values

$$\overline{s} = \prod_{i \in S} s_i \bmod N.$$

– *Correction*: The β_i values will be used to find the exponent which will be used to correct the incomplete decryptor. Compute the incomplete public key $\overline{\beta}$ as

$$\overline{\beta} = \prod_{i \in S} \beta_i \bmod N. \tag{5.12}$$

Let $\kappa_s = {c_1}^{M_S} \bmod N$ and $\kappa_\beta = g^{-M_S} \bmod N$ be the *correctors* for s and β, respectively. The corrector exponent δ is obtained by trying

$$\overline{\beta}\kappa_\beta^j \stackrel{?}{\equiv} \beta \pmod N \tag{5.13}$$

for $0 \leq j < t$.

– *Extracting the message*: Compute the message w as

$$s = \overline{s}{\kappa_s}^{\delta} \bmod N,$$
$$w = sc_2 \bmod N.$$

where δ denotes the value of j that satisfies (5.13).

As in the case of RSA, the decryptor $\overline{s}$ is *incomplete* since we need to obtain $y = \sum_{i \in S} u_i \bmod M_S$ as the exponent of c_1^{-1}. Once this is achieved, $(c_1^{-1})^y \equiv (c_1^{-1})^\alpha \pmod N$ since $y = \alpha + 2Ap'q'$ for some A.

When the equality in (5.13) holds we know that $\beta = g^\alpha \bmod N$ is the correct public key. This equality must hold for one j value, denoted by δ, in the given interval since the u_i values in (5.10) and (5.11) are first reduced modulo M_S. So, combining t of them will give $\alpha + am_0 + \delta M_S$ in the exponent in (5.12) for some $\delta \leq t - 1$. Thus in (5.12), we obtained

$$\overline{\beta} = g^{\alpha + am_0 + \delta M_S} \bmod N \equiv g^{\alpha + \delta M_S} = \beta g^{\delta M_S} = \beta \kappa_\beta^{-\delta} \pmod N$$

and for $j = \delta$ equality must hold. Actually, in (5.12) and (5.13), our purpose is not to compute the public key since it is already known. We want to find the

corrector exponent δ in order to obtain s, which is equal to the one used to obtain β. This equality can be seen as follows:

$$\begin{aligned} s \equiv {c_1}^{-\alpha} &= \beta^{-r} \\ &= \left(g^{-(\alpha+(\delta-\delta)M_{\mathcal{S}})}\right)^r \\ &= {c_1}^{-(\alpha+am_0+\delta M_S)} \left({c_1}^{M_S}\right)^{\delta} = \overline{s}{\kappa_s}^{\delta} \pmod{N} \end{aligned}$$

If the ith user is honest then the proof succeeds since ${c_1'}^{D_i} {s_i}^{-\sigma_i} = {c_1'}^{r} \bmod N$ and ${g_i}^{D_i} {v_i'}^{-\sigma_i} = {g_i}^{r} \bmod p_i$. The soundness property can be proved with a proof similar to the one in Section 5.3.1.1. Note that $\gcd(N, p_i) = 1$ for all users and $\lambda(N) = 2p'q'$ is secret. A similar proof can be given for the zero knowledge simulatability as the one in Section 5.3.1.1. We omit the security proof here since the structure of the simulator is very similar to the one in Theorem 1 of Section 5.3.1.1.

Chapter 6

Conclusion

In this thesis, we proposed CRT-based secret and function sharing schemes.

Regarding the secret sharing problem, we proposed verifiable and proactive secret sharing schemes based on the CRT. We also showed how to modify the Asmuth-Bloom SSS to use it for various problems. We proved that these modifications do not disrupt the perfectness of the scheme.

Regarding the function sharing problem, robust sharing of the RSA signature/encryption, the ElGamal and Paillier decryption and the DSS signature functions with the Asmuth-Bloom SSS are investigated. Previous solutions for sharing these functions were traditionally based on the Shamir's and Blakley's SSSs [21, 22, 23, 29, 34, 50, 66, 68]. To the best of our knowledge, the schemes described in this thesis are the first secure FSSs that use the Asmuth-Bloom SSS.

For some cases, the users have some private data $y_1, y_2, \cdots, y_n$ and they want to compute the value of a public function without revealing their private data. Secure multiparty computation (MPC) deals with this problem. This problem was first proposed by Yao [73]. In his paper, he proposed the millionaire problem in which Alice and Bob are two millionaires who want to find out which one is richer without revealing their wealth. The solutions proposed for the MPC problem are usually based on the secret sharing schemes, mostly on Shamir's SSS.

Using CRT-based SSSs for MPC schemes can be investigated as future research.

Bibliography

[1] M. Abdalla, O. Chevassut, P.-A. Fouque, and D. Pointcheval. A simple threshold authenticated key exchange from short secrets. In *Proc. of ASIACRYPT 2005*, volume 3778 of *LNCS*, pages 566–584. Springer-Verlag, 2005.

[2] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Trans. Information Theory*, 29(2):208–210, 1983.

[3] F. Bao, R. H. Deng, and H. Zhu. Variations of Diffie-Hellman assumption. In *Proc. of ICICS 2003*, volume 2836 of *LNCS*, pages 301–312. Springer-Verlag, 2003.

[4] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In *Proc. of PODC 2001, 20th ACM Symposium on Principles of Distributed Computing*, pages 274–283, 2001.

[5] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[6] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Proc. of EUROCRYPT 1994*, volume 950 of *LNCS*, pages 92–111. Springer-Verlag, 1994.

[7] J. C. Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *Proc. of CRYPTO'86*, pages 251–260, London, UK, 1987. Springer-Verlag.

[8] J. C. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *Proc. of CRYPTO'88*, volume 403, pages 27–35, London, UK, 1990. Springer-Verlag.

[9] G. Blakley. Safeguarding cryptographic keys. In *Proc. of AFIPS National Computer Conference*, 1979.

[10] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. *J. ACM*, 48(4):702–722, 2001.

[11] F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT'2000: Advances in Cryptology*, volume 1807, pages 431–444. Springer-Berlin, 2000.

[12] F. Boudot and J. Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In *Proc. of ICICS'99*, volume 1726 of *LNCS*, pages 87–102. Springer-Verlag, 1999.

[13] I. N. Bozkurt. Master's thesis, Department of Computer Engineering, Bilkent University, Ankara, Turkey, In Progress 2009.

[14] Z. Cao and L. Liu. Boudot's range-bounded commitment scheme revisited. In *ICICS'07: Proc. of the 9th International Conference on Information and Communications Security*, pages 230–238, 2007.

[15] D. Chaum, J. H. Evertse, and J. V. D. Graaf. An improved protocol for demonstrating possesion of discrete logarithm and some generalizations. In *Proc. of EUROCRYPT'87*, volume 304 of *LNCS*, pages 127–141. Springer-Verlag, 1988.

[16] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Proc. of CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1992.

[17] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneous broadcast. In *Proc. of IEEE Focs*, pages 335–344, 1985.

[18] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.

[19] P. D'Arco and D. Stinson. On unconditionally secure robust distributed key distribution centers. In *ASIACRYPT'02*, volume LNCS 2501, pages 346–363, London, UK, 2002. Springer-Verlag.

[20] Y. Desmedt. Society and group oriented cryptography. In *Proc. of CRYPTO'87*, volume 293 of *LNCS*, pages 120–127. Springer-Verlag, 1988.

[21] Y. Desmedt. Some recent research aspects of threshold cryptography. In *Proc. of ISW '97, 1st International Information Security Workshop*, volume 1196 of *LNCS*, pages 158–173. Springer-Verlag, 1997.

[22] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Proc. of CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer-Verlag, 1990.

[23] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In *Proc. of CRYPTO'91*, volume 576 of *LNCS*, pages 457–469. Springer-Verlag, 1992.

[24] Y. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics*, 7(4):667–679, 1994.

[25] C. Ding, D. Pei, and A. Salomaa. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. World Scientific, 1996.

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

[27] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS'87: Proc. of the IEEE Symposium on Foundations of Computer Science*, pages 427–437. IEEE, 1987.

[28] M. Fitzi, J. Gray, S. Gollakota, C. P. Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In *Proc. of 3rd Theory of Cryptography Conference*, pages 329–342, 2006.

[29] P. A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Proc. of FC 2000, 4th International Conference*

on Financial Cryptography, volume 1962 of *LNCS*, pages 90–104. Springer-Verlag, 2001.

[30] Y. Frankel, P. D. MacKenzie, and M. Yung. Robust efficient distributed RSA-Key generation. In *The Thirtieth Annual ACM Symposium on Theory of Computing – STOC'98*, pages 663–672. ACM Press, New York, 1998.

[31] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO'97: Proc. of the 17th Annual International Cryptology Conference on Advances in Cryptology*, volume LNCS 1294, pages 16–30, London, UK, 1997. Springer-Verlag.

[32] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of 33rd ACM Symposium of Theory of Computing*, pages 580–589, 2001.

[33] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *CRYPTO'96*, volume LNCS 1109, pages 157–172, London, UK, 1996. Springer-Verlag.

[34] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *EUROCRYPT'96*, volume LNCS 1070, pages 354–371, London, UK, 1996. Springer-Verlag.

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

[36] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Information and Computation*, 164(1):54–84, 2001.

[37] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game? In *Proc. of 19th ACM Symposium of Theory of Computing*, ACM, pages 218–229, 1987.

[38] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *CCS'97 - Computer and Communication Security*, pages 100–110. ACM, 1997.

[39] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *CRYPTO'95*, volume LNCS 963, pages 339–352, London, UK, 1995. Springer-Verlag.

[40] S. Iftene. Secret sharing schemes with applications in security protocols. Technical report, University Alexandru Ioan Cuza of Iaşi, Faculty of Computer Science, 2007.

[41] I. Ingemarsson and G. J. Simmons. A protocol to set up shared secret schemes without the assistance of a mutually trusted party. In *EUROCRYPT'91: Advances in Cryptology*, pages 266–282. Springer-Verlag, 1990.

[42] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *IEEE Globecom'87*, pages 99–102. IEEE, 1987.

[43] K. Kaya, B. G. Dündar, S. Kalkan, and A. A. Selçuk. Threshold Paillier and Naccache-Stern cryptosystems based on Asmuth-Bloom secret sharing. In *Proc. of 1st National Cryptology Symposium*, 2006.

[44] K. Kaya and A. A. Selçuk. Threshold cryptography based on Asmuth-Bloom secret sharing. *Information Sciences*, 177(19):4148–4160, 2007.

[45] K. Kaya and A. A. Selçuk. Robust threshold schemes based on the Chinese Remainder Theorem. In *Proc. of AfricaCrypt 2008*, LNCS. Springer-Verlag, 2008.

[46] K. Kaya and A. A. Selçuk. A verifiable secret sharing scheme based on the chinese remainder theorem. In *Proc. of INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 414–425. Springer-Verlag, 2008.

[47] K. Kaya and A. A. Selçuk. Secret sharing extensions based on the Chinese Remainder Theorem. *submitted to Information Sciences*, 2009.

[48] K. Kaya and A. A. Selçuk. Sharing DSS by the Chinese Remainder Theorem. *submitted to Designs, Codes and Cryptography*, 2009.

[49] K. Kaya, A. A. Selçuk, and Z. Tezcan. Threshold cryptography based on Asmuth-Bloom secret sharing. In *Proc. of ISCIS 2006*, LNCS. Springer-Verlag, 2006.

[50] A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *Proc. of ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 331–350. Springer-Verlag, 2001.

[51] P. MacKenzie and M. K. Reiter. Two-party generation of DSA signatures. *International Journal of Information Security*, 2(3):218–239, 2004.

[52] K. M. Martin, J. Pieprzyk, R. Safavi-Naini, and H. Wang. Changing thresholds in the absence of secure channels. In *Proc. of ACISP'99*, volume 1587 of *LNCS*, pages 177–191. Springer-Verlag, 1999.

[53] M. Mignotte. How to share a secret? In *Proc. of the Workshop on Cryptography*, pages 371–375. Springer-Verlag, 1983.

[54] D. Naccache and J. Stern. A new public-key cryptosystem. In *Proc. of EUROCRYPT'97*, volume 1233 of *LNCS*, pages 27–36. Springer-Verlag, 1997.

[55] V. Nikov, S. Nikova, B. Preneel, and J. Vandewalle. Applying general access structure to proactive secret sharing schemes. In *Proc. 23rd Symposium on Information Theory in the Benelux*, pages 197–206. Springer-Verlag, 2002.

[56] R. Ostrovsky and W. Skeith. Private searching on streaming data. In *Proc. of CRYPTO'05*, volume 3621 of *LNCS*, pages 223–240. Springer-Verlag, 2005.

[57] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Proc. of EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.

[58] T. P. Pedersen. Distributed provers with applications to undeniable signatures. In *EUROCRYPT'91: Advances in Cryptology*, pages 221–242. Springer-Verlag, 1991.

[59] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91: Proc. of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK, 1992. Springer-Verlag.

[60] B. Pfitzmann and A. Sadeghi. Anonymous fingerprinting with direct non-repudiation. In *Proc. of ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 401–414. Springer-Verlag, 2000.

[61] G. Poupard and J. Stern. Security analysis of a practical on the fly authentication and signature generation. In *Proc. of EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 422–436. Springer-Verlag, 1998.

[62] L. Qiong, W. Zhifang, N. Xiamu, and S. Shenghe. A non-interactive modular verifiable secret sharing scheme. In *ICCCAS'05: Proc. of the International Conference on Communications, Circuits and Systems*, pages 84–87. IEEE, 2005.

[63] M. Quisquater, B. Preneel, and J. Vandewalle. On the security of the threshold scheme based on the Chinese Remainder Theorem. In *PKC'02: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, volume 2274 of *LNCS*, pages 199–210. Springer-Verlag, 2002.

[64] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of 21st ACM Symposium of Theory of Computing*, pages 73–85, 1989.

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

[66] A. D. Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely? In *Proc. of STOC'94*, pages 522–533. ACM, 1994.

[67] A. Shamir. How to share a secret? *Comm. ACM*, 22(11):612–613, 1979.

[68] V. Shoup. Practical threshold signatures. In *Proc. of EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer-Verlag, 2000.

[69] M. Stadler. Publicly verifiable secret sharing. In *Proc. of EUROCRYPT'96*, pages 190–199. Springer-Verlag, 1996.

[70] R. Steinfeld, J. Pieprzyk, and H. Wang. Lattice-based threshold changeability for standard CRT secret-sharing schemes. *Finite Fields and Their Applications*, 12:653–680, 2006.

[71] R. Steinfeld, J. Pieprzyk, and H. Wang. Lattice-based threshold changeability for standard Shamir secret-sharing schemes. *IEEE Trans. of Information Theory*, 53:2542–2559, 2007.

[72] W. Wei, T. Trung, S. Magliveras, and F. Hoffman. Cryptographic primitives based on groups of hidden order. *Tatra Mountains Mathematical Publications*, 29:147–155, 2004.

[73] A. C. Yao. Protocols for secure computations. In *Proc. of FOCS'82*, pages 160–164. IEEE, 1982.