

國立交通大學

資訊工程學系

博士論文

金鑰演化密碼學之研究



A Study of Key-Evolving Cryptosystems

研究生：曾志嘉

指導教授：曾文貴教授

中華民國九十五年七月

金鑰演化密碼學之研究
A Study of Key-Evolving
Cryptosystems

研究生：曾志嘉

Student: Zhi-Jia Tzeng

指導教授：曾文貴 博士

Advisor: Wen-Guey Tzeng

國立交通大學資訊學院

資訊工程學系

博士論文



A dissertation is submitted to
Department of Computer Science
National Chiao Tung University
for the degree of
doctor of philosophy

in

Computer Science

July 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年七月

金鑰演化密碼學之研究

研究生：曾志嘉

指導教授：曾文貴 博士

國立交通大學 資訊學院 資訊工程學系

摘要

在公開金鑰密碼學裡，許多論文已經討論過了金鑰洩露的處理問題，有一些方法被用來處理金鑰洩露的問題，比如分散式的門檻方式、預防的機制以及智慧卡的硬體保護方法等。在這篇論文中，我們提出第一個金鑰演化的密碼方法去處理這個問題，它如同前向安全的簽章 (forward-secure signature) 系統一樣，有一個特性：就是私密金鑰會隨著時間而改變，但是公開的金鑰卻是固定不變的。我們將金鑰的生命週期分成小的時間區段，在時間區段 j ，解密者擁有時間區段 j 的私密金鑰 SK_j ；在時間區段 $j+1$ ，解密者擁有時間區段 $j+1$ 的私密金鑰 SK_{j+1} 。但是在金鑰的有效期間內，公開金鑰卻是固定不動的。如果一個傳送者要送訊息 m 給解密者，他必須由公開金鑰 PK 計算時間區段 j 的公開金鑰 PK_j ，再將信息加密成 $\langle j, c \rangle$ 。當時間區段從 j 移轉到 $j+1$ ，解密者需要更新他的私密金鑰 SK_j 成為 SK_{j+1} ，然後立刻刪除 SK_j 。金鑰演化的密碼系統，即使私密金鑰 SK_j 遺失或是洩露，不會影響其他時段加密的訊息之安全。這篇論文主要的結果如下：

1. 我們提出三個簡單的金鑰演化公開金鑰加密方法，這些方法具有 ϵ -彈性的性質，使得 ϵ 把私密金鑰被暴露，仍然不影響其他時間區段的加密訊息的安全。和傳統的公開金鑰密碼系統比較，新的加密方法的密文中包含了時間的資訊。為了證明新的密碼方法是安全的，我們假設 DDH 問題是難的，和 random oracle 模式成立。我們的新的密碼方法是可以抵擋被動的攻擊 (passive attack) 和適當的選擇密文攻擊法 (adaptive chosen ciphertext attack)。
2. 我們提出解密者如何在 TA 們的幫助下，使用安全的分散式計算方法，算出新的私密金鑰。然後提出分散式的金鑰演化密碼系統，並且討論如何將分散式的金鑰演化密碼系統，和預防的機制 (proactive mechanism) 作結合，來加強 TA 的安全性。
3. 我們提出一個分散式門檻的前向安全簽章方法，加強了 Abdalla and Reyzin 的前向安全簽章方法的安全性，主要是藉由兩個技巧，門檻的方式和預防的機制來

做到的。這個分散式門檻的前向安全簽章方法組合了多項式秘密分享 (polynomial secret sharing) 和乘法的 (multiplicative) 技巧。如果原來的單一使用者 (single-user) 的簽章方法是安全的，我們可以證明新的分散式的簽章方法是安全的。

金鑰演化加密系統可以應用到以憑證為基礎的認證系統，使得認證協定達到前向安全 (forward-secure) 和後向安全 (backward-secure)，並且可以減少 CRL 的儲存代價。公開金鑰的憑證被應用在電子商務、存取網際網路資源及個人通訊服務等方面，為了儲存被取消的憑證，網路服務供應者 SP (Service Provider) 需要額外的儲存空間。而這樣的系統安全是和目前時段的秘密金鑰有關，既然時段 j 的私密金鑰 SK_j 將在時段 $j+1$ 時無效，所以時段 j 的 CRL，在時段 $j+1$ 可以不需要存著。所以每一個新的時段的開始，CRL 的大小就是 0，因而可以節省儲存 CRL 的空間，減少 SP 對儲存 CRL 所付出的代價。另外我們討論了“同步時間”的問題，我們假設一個服務供應者存在著同步時間的伺服器。

關鍵字：金鑰演化密碼系統， k -彈性，公開金鑰加密，前向安全，門檻密碼方法，CRLs，憑證，認證，同步時間。



A Study of Key Evolving Cryptosystems

Student: Zhi-Jia Tzeng

Advisor: Wen-Guey Tzeng

Department of Computer Science
College of Computer Science
National Chiao Tung University



Abstract

The key exposure problem of public key encryption schemes has been discussed in the open literature. Threshold cryptosystems, proactive mechanism and smart card are used for many years to handle this problem. In this thesis, we propose the first key-evolving paradigm to deal with the key exposure problem. The key-evolving paradigm is like the one used for forward-secure digital signature schemes. Let the lifetime of the master secret key be divided into time periods such that at time period j , the decryptor holds the private key SK_j , while the public key PK is fixed during its lifetime. At time period j , a sender encrypts a message m as $\langle j, c \rangle$, which can be decrypted only with the private key SK_j . When the time makes a transit from period j to $j + 1$, the decryptor updates its private key from SK_j to SK_{j+1} and deletes SK_j immediately. The key-evolving paradigm assures that compromise of the private key SK_j does not jeopardize the message encrypted at the other time periods. Our results are listed in the following.

1. We propose three simple key-evolving public key encryption schemes with z -resilience such that compromise of z private keys does not affect confidentiality of messages encrypted in other time periods. Comparison to the public key cryptosystems, a ciphertext in the new scheme contains time information. Assuming that the decisional Diffie-Hellman

(DDH) problem is hard and the random oracle model, we show that our schemes are secure against passive adversaries and against adaptive chosen ciphertext attack.

2. We present how key-evolving with TAs does. The decryptor can evolve the private key by the aid of TAs in a secure distributed way. Then, we consider the case of distributed key-evolving encryption scheme. Furthermore, we combine the distributed methods with proactive mechanism to enhance the security of TAs.
3. We propose a distributed forward-secure signature to enhance the security of Abdalla and Reyzin's forward-secure signature scheme via threshold and proactive mechanisms. Our distributed threshold forward-secure signature scheme combine both multiplicative and polynomial secret sharing tricks. Then, We can prove that our scheme is secure if the single-user scheme is secure.

The key-evolving public key encryption schemes are applied to reduce the storage cost of Certificate Revocation Lists (CRLs) in the encryption certificate-based authentication protocols. Public key certificates have been used in many applications, such as electronic commerce, accessing Internet resources and personal communications services, etc. Let Service Provider (SP) provide some services. The users subscribe the services from SP. While accessing the services, SP authenticates the identity of the user via a certificated authentication protocol based on the key-evolving public key encryption scheme. However, if a user's secret key for the certificate is lost or compromised, SP need additional storage cost for saving CRLs. The security of such certificate-based protocols depends on the secret key of the current time period. Since the disclosed secret key SK_j of time period j is automatically revoked at time period $j + 1$, CRLs of time period j does not be maintained at time period $j + 1$. That is, in the beginning of a new time period the size of CRLs is reset to zero. Thus, the size of CRLs can be reduced. Finally, we discuss the problem of time synchronization. Our schemes assume that SP should have time server for synchronization among SP and all subscribers.

Keywords: key-evolving, k -resilience, public key encryption, forward security, threshold, CRLs, certificate, authentication, time synchronization.

誌謝

首先將論文獻給往生的父親及被「活摘器官」中離開人世的法輪功朋友，表達對他們的思念與敬意，願他們在天之靈護佑我們走向成功之路，就像這本論文的出世，也是歷盡艱難阻隔，需要去克服這些困難。

最感謝的就是我的指導教授曾文貴老師，除了在指導學習上付出許多的心血；在精神上，老師也承受了許多，常常需要擔心我夠不夠用心，有沒有準備齊全。在我口試時，老師及口試委員們給我諸多意見，與包容我的不足，讓我的論文更臻於完善。所以在此對口試委員一併致謝。

再來要感謝母親多年來的包容，常年隻身在外無法替母親分憂，而自己的身體狀況，則是最令母親擔心的。還有家人的關心與鼓勵，成為我精進的動力。

接著要感謝李洪志先生，傳出的「法輪功」讓我得到健康。在我健康狀況最差的時候，人生最低迷、想要放棄的時候，李洪志先生傳出的功法，讓我重拾健康與信心，讓我有勇氣繼續完成學業。

還有感謝我的同事碧月和毓秀學姊，他們督促、鼓勵與幫助，讓我教書可以順利，也讓我學習無後顧之憂。還有櫻花學姊，她介紹我生機飲食與參加法輪功教師研習，讓我生命有了轉折。

最後感謝小翠及實驗室的成員：成康、智明、孝盈、阿田和碩士班的學弟，他們的幫忙讓我口試能夠順利，論文能夠完成。

要感謝的人太多了，在此對所有幫助與關心過我的親友一併致謝！謝謝！

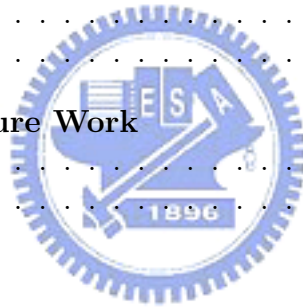
曾志嘉 2006/07/27



Contents

中文摘要	i
英文摘要	iii
誌謝	v
目錄	vi
圖目錄	ix
1 Introduction	1
1.1 Motivations	1
1.2 Contribution of this thesis	3
1.3 Related works	4
1.3.1 Forward-secure signature schemes	4
1.3.2 Key-evolving encryption schemes	7
1.3.3 Threshold cryptography	8
1.3.4 Proactive security	9
1.3.5 Certificate Revocation List	9
1.3.6 Time synchronization	10
2 Preliminaries and Security Models	13
2.1 Hardness assumptions	13
2.2 Security models	15
2.2.1 Security against passive adversaries	15
2.2.2 Security against adaptive chosen ciphertext attack	16
2.3 Definition	17
3 Key-Evolving Public Key Encryption Schemes	21
3.1 KE-ENC against passive adversaries	21
3.1.1 Security analysis	23
3.2 KE-ENC against adaptive chosen ciphertext attack	25
3.2.1 Under the random oracle model	25
3.2.2 Security analysis	26
3.2.3 Under the standard model	29
3.2.4 Security analysis	30

4	Distributed and Proactive Key-Evolving Encryption	41
4.1	Key evolving with TA	41
4.1.1	Distributing TA's secret	41
4.1.2	Security analysis	43
4.1.3	Proactivizing TA's shares	44
4.1.4	Security analysis	45
4.2	Distributed KE-ENC schemes	46
4.3	Distributed KE-ENC with proactive security	47
5	Threshold Forward-Secure Signature Schemes	49
5.1	Building blocks	51
5.2	Threshold forward-secure signature scheme	57
5.3	Security analysis	59
5.4	Discussion	64
6	Applications to key-evolving public key certificate-based authentication protocol	65
6.1	Key-evolving public key certificate-based authentication protocol	67
6.2	Extension	69
6.3	Security	69
6.4	Implementation	69
7	Conclusion and Future Work	73
7.1	Conclusion	73
7.2	Future work	74
	Bibliography	75
	Appendix	89



List of Figures

3.1	KEENCBASIC(part1) – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against passive adversaries.	23
3.2	KEENCBASIC(part2) – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against passive adversaries.	24
3.3	KEENCROM – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against the adaptive chosen ciphertext attack under the random oracle model.	38
3.4	KEENCSTM (part 1)– discrete logarithm based key-evolving scheme with z -resilience and semantic security against the adaptive chosen ciphertext attack under the standard model.	39
3.5	KEENCSTM (part 2) – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against the adaptive chosen ciphertext attack under the standard model.	40
6.1	A key-evolving certificate-based authentication protocol	67
7.1	Bellare and Miner’s forward-secure signature scheme is based on the hardness of the square root problem. (part 1)	90
7.2	Bellare and Miner’s forward-secure signature scheme is based on the hardness of the square root problem. (part 2)	91
7.3	Abdalla and Reyzin’s forward-secure signature scheme is based the hardness of the 2^l -th root problem.	94
7.4	Itkis and Reyzin’s forward-secure signature scheme is based on GQ signature.(part 1)	95
7.5	Itkis and Reyzin’s forward-secure signature scheme is based on GQ signature.(part 2)	96
7.6	Canetti et al’s binary tree encryption scheme is based on bilinear Diffie-Hellman assumption.(part 1)	97
7.7	Canetti et al’s binary tree encryption scheme is based on bilinear Diffie-Hellman assumption.(part 2)	98

Chapter 1

Introduction

1.1 Motivations

In a public-key cryptosystem the secret key is used to decrypt or sign messages until the lifetime of the secret key is over. The security of public-key cryptosystems is based on the security of the secret key. Once the secret key is disclosed or compromised, the adversary behaving as the key owner can sign or decrypt any messages in the past. The fact that the secret key is fixed during the lifetime is the risk of the current public-key cryptosystem. An adversary can record the history of sent messages and then get useful messages after getting the secret key. For example, the sender Alice sends a message m to the decryptor Bob with Bob's public key PK . Although an attacker Carol does not know Bob's private key SK at present time, he can eavesdrop and record the ciphertext $c = E(PK, m)$. Later on, Carol manages to get Bob's private key SK , she can get the message m no matter how much time has elapsed. The message m may carry information that is useful for a long period of time. There are several ways to resolve the problem. One way is that Bob replaces his public key when his private key is exposed. But, in this case every user has to update Bob's public key in its database when Bob upgrades his public key. This is quite costly. Furthermore, it may not be practical since Bob may not be aware of the disclosure of his private key. Another way is to protect Bob's

private key via a smart card so that key exposure is not possible. Another way is to share SK to n trusted agents (TA's) in a k -out-of- n threshold scheme such that at least k trusted agents can fully recover SK. After receiving a ciphertext c , Bob uses the TA's to decrypt c in a distributed way. To protect TA's shares of SK for a long time, we can further proactivize TA's shares. In this case TA's not only bear heavy load of computation, but also should stay on-line always to provide decryption service. There are of course other solutions. In the thesis, we propose the key-evolving public-key encryption scheme to handle the key exposure of the public-key encryption scheme. We concentrate on designing key-evolving public-key encryption schemes.

The concept of key-evolving encryption schemes comes from the forward-secure digital signature scheme. In the key-evolving public-key cryptosystem the time is divided into time periods. When the time period transfers from j to $j + 1$, the secret key SK_j is updated to SK_{j+1} . Given SK_j , one can not derive the previous secret key SK_{j-1} . Thus, an adversary who obtains the key of the time period j does not compute any key before the time period j . Therefore, the messages before the time period j is not disclosed even though an adversary gets the secret key of the time period j , SK_j . The key-evolving cryptosystem is more secure in the sense that it allows many times of key exposure. In our encryption schemes, we assume that there exists TA. Thus, a decryptor can evolve his secret key by the aid of the TA. The TA may be a secure device.

We further enhance the security of the key-evolving encryption scheme by considering that the threshold key-evolving cryptosystem. The threshold key-evolving cryptosystem is that the secret key is shared to n servers in a distributed way such that t out of them can recover the secret key. However, less than t servers can not obtain any message about the secret key. Furthermore, we combine the threshold cryptosystem with proactive security to protect the secret key. At first, the secret key is shared to a group of servers in a t -out-of- n scheme. Each server has a share. When the time makes a transit, the secret

key and the shares held by the servers are updated. Thus, if an adversary obtains less than t shares, it cannot derive the secret key and compute new shares at a new time period. These shares held by the adversary will become invalid at the new time period.

1.2 Contribution of this thesis

In the thesis, we propose the first key-evolving paradigm to deal with the key exposure problem of public-key encryption schemes. We present how key-evolving with TAs does. Then, we describe our threshold forward-secure signature scheme based on the forward-secure signature scheme of Abadalla and Reyzin [4]. Finally, we show an application for the key-evolving encryption scheme. In the following, we summarize our results.

1. We propose three simple key-evolving encryption schemes that are z -resilient public-key encryption schemes such that z times of key exposures do not release any information pertinent to ciphertexts that are encrypted with non-exposed private keys. We assume that there exists a secure device (or TA). The basic key-evolving encryption scheme is semantically secure against passive adversaries. Furthermore, we modify the basic scheme to achieve semantically secure against the chosen ciphertext attack under the random oracle model. Another modification of the basic scheme achieves semantically secure against the chosen ciphertext attack under the standard model. The size of a public key is independent of the total number of time periods, but dependent on resilience. On the other hand, the size of a private key is a constant. Without counting the pre-computation time for each time period, both encryption and decryption operations take 2 modular exponentiations. The pre-computation time is independent of time period, but dependent on resilience.
2. We present how key-evolving with TAs does. The decryptor can evolve

the secret key by the aid of TAs in a secure distributed way. We also consider the case of distributed key-evolving encryption scheme. Furthermore, we combine the distributed methods with proactive mechanism to enhance the security of TAs.

3. We also propose a threshold forward-secure signature scheme based on the forward-secure signature scheme of Abadalla and Reyzin [4]. The security of the threshold forward-secure signature scheme is based on the 2^l -th root problem. We show that the security of the forward-secure threshold scheme based on that of the single-user scheme.
4. We propose a key-evolving encryption certificate-based authentication protocol. In the encryption certificate-based authentication protocol, the key-evolving encryption scheme can make the authentication protocol achieve forward and backward secrecy. To revoke a subscriber, CRL is used to save the certificate of the revoked subscriber. In such a protocol, the storage cost of CRLs can be reduced when times make a transit from j to $j + 1$. Since CRLs of the time period j does not maintain at the time period $j + 1$, the size of the CRLs is reset to zero at the beginning of time period $j + 1$. Thus, the storage cost is reduced. In addition, we discuss time synchronization in our application.

1.3 Related works

We survey the related results on forward-secure signature schemes, key-evolving encryption schemes, threshold cryptography, proactive security, certificate revocation list and time synchronization.

1.3.1 Forward-secure signature schemes

In traditional signature schemes a signature is independent of time. If the secret signing key is exposed, one can sign arbitrary messages. Comparison

to the traditional signature scheme, in a forward-secure signature scheme, the signing key depends on time periods. When time transits from time j to $j + 1$, the signing key is updated from SK_j to SK_{j+1} and SK_j is deleted immediately. However, the public key is the same during the entire lifetime. If an adversary gets the signing key SK_t of time period t , he can fake the signatures after the time period t , but cannot fake the signatures prior to the time period t .

Bellare and Miner [13] proposed the first forward-secure signature scheme based on difficulty of computing the square roots modulo a Blum integer. The scheme is actually converted from Fiat and Shamir's identification scheme [36]. To achieve security strength of level l , their scheme uses l public keys and l secret keys. The computation time of a signature depends on the largest time period T .

Afterward, Abdalla and Reyzin [4] proposed an improvement based on the 2^l -th root problem [43, 64, 76, 75]. With the same level of security strength, their scheme uses one public key and one secret key only. However, the computation time of a signature also depends on the large time period T .

Krawczyk [54] proposed a simple transformation from any signature scheme to a forward-secure signature scheme. They assume that a forward-secure pseudorandom generator exists and the base signature scheme (e.g., DSA, RSA) is secure. Their scheme uses an initial seed to generate all key pairs and related certificates. Therefore, the time of key generation is related the largest time period T . Their scheme uses the base signature scheme to sign a message. A signature consists of $cert_t$ and $SIGN_{SK_t}(M)$, where $SIGN_{SK_t}(M)$ is the signature of M at time period t , which is signed by the base signature scheme.

Itkis and Reyzin [49] proposed a new scheme based on GQ signature scheme [42]. The security of the scheme is based on e -th root problem. The new scheme optimizes the procedures of signing and verifying. That is, the procedures of signing and verifying is independent of the largest time period T . However, the procedure of key updating depends on T . Afterwards, Itkis

and Reyzin [50] proposed a signer-base intrusion-resilient signatures. The signature scheme combines both notions of forward-secure and key-insulated signature schemes. The user has two modules, *signer* and *home base*. Their scheme remains secure if the intruder does not compromise of both modules simultaneously. The signer-base intrusion-resilient signatures are still based on GQ signature. Their schemes are provably secure in the random oracle model based on the strong RSA assumption. Nicolás et al. [74] proposed a $(N - 1, N)$ -key-insulated signature scheme. This scheme is more efficient than previous proposals and whose key length is constant and independent of the number of time periods. The security of the scheme also is based on the strong RSA assumption in the random oracle model. The strong RSA assumption is: given a number n that is the product of two prime numbers and a value γ , it is computationally infeasible to find $\beta \in Z_n^*$ and $v > 1$ such that $\beta^v = \gamma \pmod n$. More key-insulated signature scheme have proposed in the literature [25, 26]. A hierarchical key-insulated signature scheme is discussed in the literature [60]. Kim and Kim [55] proposed an intrusion-resilient key-evolving Schnorr signature. In their scheme, if secret keys of all periods are not compromised, it is not possible to forge signatures relating to non-exposed secret keys. More forward-secure signature schemes have proposed in much literature [1, 52, 69].

In addition, Bellare et al.[15] constructed forward-secure pseudorandom number generators and added forward security to the private-key cryptosystems. They give a rigorous analysis of a forward-secure pseudorandom number generator.

Abadalla et al. has proactivized the Bellare-Miner forward-secure signature scheme [3]. They proposed two threshold signature schemes in proactivizing Bellare-Miner forward-secure signature scheme. One scheme uses multiplicative secret sharing and the other uses polynomial secret sharing. We propose a robust forward-secure digital signature based on the scheme of Abadalla and Reyzin (see chapter 5). Our scheme uses the both techniques of the multiplicative secret sharing and the polynomial secret sharing.

1.3.2 Key-evolving encryption schemes

The concept of key-evolving encryption schemes comes from the forward-secure digital signature scheme. Concurrently, Dodis et al. proposed the notion of key-insulated security [25]. They constructed a (t, N) -key-insulated encryption scheme based on any standard public-key encryption scheme. Their schemes achieve forward and backward security under the disclosure of t keys at most. The secret key stored on the insecure device is updated at discrete time period via the help of the physically-secure device. Under the DDH assumption, they proposed a semantically-secure key-updating encryption scheme. Then, they modify the basic scheme to be a chosen-ciphertext-secure key-updating encryption scheme. We describe their scheme as follows. Let G_q denote a group with a prime order q . Let g, h be the generators in G_q . The decryptor selects two polynomials with degree t . The public key contains g, h and Pedersen commitments [81] $\{z_0^*, \dots, z_t^*\}$ to the coefficients of the two polynomials. Let the coefficients of the two polynomials be $\{x_0^*, \dots, x_t^*\}$ and $\{y_0^*, \dots, y_t^*\}$. The decryptor holds the secret $\{x_0^*, y_0^*\}$ and the remaining coefficients are saved by the secure device. To encrypt M during time period i , the encryptor computes the public key $z_i = \prod_{j=0}^t (z_j^*)^{i^j}$, and then computes a ciphertext $C = \langle i, (g^a, h^a, z_i^a M) \rangle$ for a random $a \in Z_q$. Since $z_i = g^{f_x(i)} h^{f_y(i)}$, the decryptor who has the secret key $SK_i = (f_x(i), f_y(i))$ can decrypt C to obtain M . As for key evolution, the device transmits partial key $SK'_i = (x'_i, y'_i)$ to the decryptor at time period i . Note that $x'_i = f_x(i) - f_x(i-1)$ and $y'_i = f_y(i) - f_y(i-1)$. Since the decryptor has SK_{i-1} , he can compute SK_i . At this point, the decryptor erases SK_{i-1} . In addition, Bellare and Palacio proposed a key-insulated encryption with optimal threshold [14].

Canetti et al. proposed a forward-secure public-key encryption scheme based on the bilinear Diffie-Hellman assumption in the random oracle model [21]. The bilinear Diffie-Hellman (BDH) problem is formalized by Boneh and Franklin [10]. We describe it briefly as follows. Given $(\mathbf{G}_1, \mathbf{G}_2, \hat{e})$ for random $P, aP, bP, cP \in \mathbf{G}_1$, no probabilistic polynomial time algorithm for com-

puting $\hat{e}(P, P)^{abc}$. Their forward-secure encryption schemes are constructed by any BTE(binary tree encryption) scheme. Their scheme are based on the hierarchical identity-based encryption scheme as well [8, 45, 48, 87, 97].

Lu and Shieh [59] consider the key-evolving protocols in the secret-key setting. They concentrate on the security and efficiency of key-evolving protocols. They proposed two key-evolving protocols. One protocol uses the Feldman's technique. The other uses the Manurer-Yacobi scheme [72]. Manurer and Yacobi showed that every square modulo has a discrete logarithm to the base of g . The knowledge of factoring of n is the trapdoor for solving the discrete logarithm problem; that is, given PK_i , with the trapdoor knowledge of the factoring n , he can compute $SK_i = \log_g(PK_i)$, where $PK_i \in QR_n$. However, they only consider the case that the adversary cannot fully determine an un-exposed secret key.

1.3.3 Threshold cryptography

Consider the situation that a centralized system is attacked such that the master secret is disclosed. This results in that the system manager must pay the quite cost for reconstructing the system or server. The centralized scheme is less flexible than the distributed one since only an adversary who successfully controls the system once or one server can destroy the system. To avoid the single server's failure, we use the distributed system with multiple servers to replace the centralized system with one server. A familiar distributed scheme is a t -out-of- n threshold scheme. Threshold scheme has the spirit of secret sharing [6, 88], where the shared secret is distributed to several servers such that each server owns a share. In a t -out-of- n scheme, at least t servers can recover the shared secret but less than t servers can not obtain any information of the secret key. For security, an adversary cannot obtain any message if he controls less than t servers. Threshold cryptosystems have proposed in much literature [24, 37, 51, 80].

1.3.4 Proactive security

Ostrovsky and Yung [77] first proposed security and availability in the presence of a mobile adversary. To secure long-lived cryptographic keys that cannot be replaced easily, proactive mechanism can deal with the increasing number of threats to local and international network domains. For proactive security, the share in each party is refreshed at the end of each time period, but the signing secret key is unchanged at all time. A proactive cryptosystem remains secure as long as the adversary does not corrupt more than t parties in each time period. The shares of corrupted parties become useless when time enters the next time period. A number of very useful cryptographic functions have efficiently proactivized, e.g., pseudorandomness, secret sharing and public key schemes [19, 24, 38, 47, 46, 77]. Proactive mechanism can incorporate with threshold cryptosystems to protect against a mobile adversary. The added advantage of proactivization is that in the proactive systems an adversary has only a short period of time to break into any t -out-of- n servers, while in the long-lived threshold systems the adversary has a long time to break into any t servers. The adversary obtained any $m(< t)$ shares in the old time periods, which are invalid shares in the new time periods. Thus, proactive mechanism enhances the security of the threshold schemes. There is much literature about proactive cryptosystems [2, 20, 39, 32, 33, 85, 35].

1.3.5 Certificate Revocation List

Public key certificates are widely used to guarantee the authenticity of commercial web-sites, to secure the privacy of personal communication, or to authenticate the identity of a subscriber, etc. Certificate revocation is taken to deal with the secret key compromise or loss prior to the expiration date. The standard of certificate revocation is proposed in X.509 directory framework [96] and the Internet draft standard Public Key Infrastructure [5]. When a subscriber wants to revoke his certificate, he sends SP a revocation notice, which

is a signed message identifying the certificate to be revoked.

Storage cost and communication cost are two primary measures in the strategies of certificate revocation. Kocher's certificate revocation tree [53] is to reduce these two costs. The methods proposed by Cooper and McDaniel and Jamin [16, 68] reduces server's load. Wright et al [91] proposed a fault-tolerant method for distribution of revocations and certificate updates. The method is to reduce the communication cost. Rivest [82] proposed to eliminate CRLs and suggested a two-level staged expiration. However, this leads to the more complex system for handling the case of key compromise. Furthermore, one can limit the lifetime of certificates to eliminate CRLs, but this increases the load of server since new certificates are issued and distributed more frequently. McDaniel and Rubin [71] think that CRLs still is a necessary part of any PKI. Other more discussions for certificate revocation can be found in the literature [34, 66, 67, 73].

1.3.6 Time synchronization

Our proposals are dependent on "time". In physical applications, time synchronization is not easy. For a personal computer, neither the software or hardware clock is suitable for accurate timekeeping. If your computer can access to the Internet, you can synchronize its time clock to an Internet time server. Currently, time protocol [79], daytime protocol [78], network time protocol [63] and simple network time protocol [65] are four major timing protocols used in Internet. Many synchronization protocols [56, 62, 89] are proposed. Most of them share a basic design: a server periodically sends a message containing its current clock value to a client. However, in the ad hoc network, neither logical time [57, 61] nor classical physical clock synchronization algorithms [58, 84] can be used to solve temporal ordering and other real-time issues in such environments. Römer [83] presents a time synchronization protocol in the sparse ad hoc networks. Elson et al. [29] discussed time synchronization in low-cost and low-power devices. They proposed reference-broadcast synchro-

nization protocol in which nodes send reference beacons to their neighbors using physical-layer broadcasts. Maroti et al. [70] proposed the flooding time synchronization protocol in the wireless sensor network. Their protocol uses low communication bandwidth and it is robust against node and link failures. The flooding time synchronization protocol achieves its robustness by utilizing periodic flooding of synchronization messages, and implicit dynamic topology update. There are many protocols proposed for sensor networks in much literature [28, 40, 93].





Chapter 2

Preliminaries and Security

Models

If Alice wants to securely send messages to Bob over an insecure channel, a public-key encryption serves the purpose. In the chapter, we consider the security of a public key encryption scheme. We consider two attacks: One is a passive attack. The other is an active attack. A passive attack is that an adversary who only eavesdrops the sent messages does not actively send forged or modified messages to attack the sender or receiver. An active attack is that an adversary can forge a message, forge a signature or do whatever to make the protocol goal fail. In an active attack, we further consider the adaptive chosen ciphertext attack under the random oracle model and the standard model. We give the formal definitions of the public-key encryption schemes under various security models. In addition, we will state the cryptographic hardness assumptions related to our schemes. Finally, we give the formal definition of the key-evolving public key encryption scheme.

2.1 Hardness assumptions

We describe some cryptographic assumptions as follows.

Discrete logarithm assumption. The discrete logarithm problem is that, given a prime p , a generator g of Z_p^* , and an element $y \in Z_p^*$, finding the integer $x, 0 \leq x \leq p - 2$ satisfying $g^x \equiv y \pmod{p}$ is difficult. That is, we assume that for any probabilistic polynomial time algorithm \mathcal{A} , for any enough prime n , for any $k > 0$,

$$\Pr_{p \in S_n, g, y \in G_q} [A(y, g, p) = \log_g y \pmod{p}] \leq 1/n^k,$$

where S_n denotes the set of n -bit prime p , where $p = 2q + 1$.

The Decisional Diffie-Hellman assumption. We need a standard assumption of solving the decisional Diffie-Hellman (DDH) problem [7]. Breaking our schemes is reduced to solving the DDH problem. Let G be a group of a large prime order q . Consider the following two distribution ensembles R and D :

- $R = (g_1, g_2, u_1, u_2) \in G^4$, where g_1 and g_2 are generators of G_q ;
- $D = (g_1, g_2, u_1, u_2)$, where g_1 and g_2 are generators of G_q and $u_1 = g_1^r$ and $u_2 = g_2^r$ for $r \in Z_q$.

The *DDH problem* is to distinguish the distribution ensembles R and D . That is, we would like to find a probabilistic polynomial-time algorithm \mathcal{A} such that

$$|\Pr[\mathcal{A}(R_n) = 1] - \Pr[\mathcal{A}(D_n) = 1]| = \epsilon(n)$$

is non-negligible, where R_n and D_n are the size- n distributions of R and D , respectively.

A *trapdoor one-way permutation* is a probabilistic polynomial-time algorithm \mathcal{G} that takes as input 1^n and outputs a triple of algorithms (f, f^{-1}, d) , where f and f^{-1} are inverses of each other and deterministic polynomial-time algorithms and d is a probabilistic polynomial-time algorithm. The range of $d(1^n)$ is a subset of $\{0, 1\}^k$ and f and f^{-1} on the range of $d(1^n)$ are permutations. Furthermore, for any probabilistic polynomial-time algorithm \mathcal{A} ,

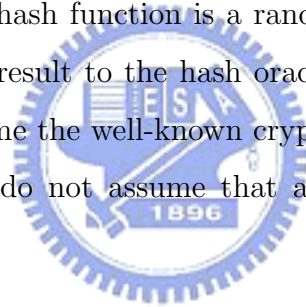
$$\epsilon(n) = \Pr[\mathcal{G}(1^n) \rightarrow (f, f^{-1}, d); d(1^n) \rightarrow x; f(x) \rightarrow y :$$

$$\mathcal{A}(f, d, y) = x]$$

is negligible. For convenience, we shall call f , not \mathcal{G} , a "trapdoor one-way" permutation.

2.2 Security models

We consider security against passive adversaries and the adaptive chosen ciphertext under the random oracle model and the standard model. An encryption scheme is secure against the passive attack if the encryption scheme achieves the semantic security. Given a public key and two different messages m_0 and m_1 , a passive adversary cannot distinguish that a ciphertext c is an encryption form of m_0 or m_1 in the polynomial time. In the random oracle model, we assume that a hash function is a random function. To compute a hash value, we query the result to the hash oracle. Comparison to the standard model, we only assume the well-known cryptographic assumptions, such as DDH assumption but do not assume that a hash function is a random function.



2.2.1 Security against passive adversaries

Assume that an adversary \mathcal{A} consists of two probabilistic polynomial-time algorithms A_1 and A_2 . A_1 takes as input a public key PK and outputs two different messages m_0 and m_1 . A_2 takes as input a public key PK and a ciphertext c of m_b ($b = 0$ or 1) and outputs a bit b' . b' is a guess of the plaintext source of c . Let KG , E and D be the key generation, encryption and decryption algorithms of a public key encryption scheme. Let PK and SK be the public and private keys, respectively.

Definition 1 *A public key encryption scheme (KG, E, D) is semantically secure against passive adversaries if a passive adversary \mathcal{A} , which is probabilistic polynomial-time, cannot distinguish the ciphertexts of any two messages m_0*

and m_1 [41]. That is, for any passive adversary \mathcal{A} ,

$$\Pr[KG(1^n) \rightarrow (PK, SK) : \mathcal{A}(PK, E(PK, m_b)) = b] = 1/2 + \epsilon(n),$$

where $\epsilon(n)$ is negligible and the probability is taken over b and the random coin tosses of KG , E and \mathcal{A} .

2.2.2 Security against adaptive chosen ciphertext attack

The adversary A has two probabilistic polynomial-time algorithms A_1 and A_2 , as described above. Comparison to the passive adversary, the active adversary can access the decryption oracle \mathcal{DO} . The attack is non-adaptive if only A_1 can access \mathcal{DO} . In the active attack, both A_1 and A_2 can access \mathcal{DO} . The adaptive chosen ciphertext attack on an encryption scheme works as follows [86]. An adversary \mathcal{A} of the attack has two probabilistic polynomial-time algorithms A_1 and A_2 . A_1 takes as input PK , makes some queries to the decryption oracle \mathcal{DO} adaptively, and outputs two messages m_0 and m_1 . Then, the encryption oracle randomly chooses a bit b and encrypts m_b as $c = E(PK, m_b)$. A_2 takes as input PK , c , m_0 and m_1 , makes some queries to the decryption oracle \mathcal{DO} in an adaptive way, and outputs b' . The decryption oracle \mathcal{DO} takes as an input a ciphertext c' , other than c , and returns its corresponding plaintext m' . If c' is invalid, \mathcal{DO} outputs '??'.

Definition 2 A public key encryption scheme (KG, E, D) is semantically secure against the adaptive chosen ciphertext attack if, for any adversary $\mathcal{A} = (A_1, A_2)$,

$$\Pr[KG(1^n) \rightarrow (PK, SK); A_1^{\mathcal{DO}}(PK) \rightarrow (m_0, m_1) : \\ A_2^{\mathcal{DO}}(PK, E(PK, m_b)) = b] = 1/2 + \epsilon(n),$$

where $\epsilon(n)$ is negligible and the probability is taken over b and all coin tosses of KG , E , A_1 and A_2 .

Random oracle model vs. Stand model

Random oracle model. The *random oracle model* assumes that hash functions used in a scheme are "truly random" hash functions. Usually, protocols using collision-resistant hash functions h are hard to be proven secure, so we assume that h behaves like a "truly random function". This is a standard approach in cryptography fields [12]. A truly random hash function takes as input a value and outputs a random value. This assumption is called as the *random oracle model*. Under the random oracle model, we cannot compute the hash value $h(v)$ of a value v , but we can query the result of $h(v)$ to the random oracle. Furthermore, since the output is random, one only can guess the correct result for an input with the probability $1/2^n$ if the output is n -bit long. Note that the queried results should be consistency, i.e., *for the same input the oracle should output the same value*. Although security under the random oracle model is not rigid, it does provide satisfactory security argument to a scheme in most cases [18]. Since there exist signature and encryption schemes which are secure in the random oracle model, but for which any implementation of the random oracle results is insecure schemes [18].

Standard model. In the standard model [23], we do not assume that the hash function h is random. Let the hash function h is a collision-resistant hash function. The security in the standard model is based on the well-known cryptographic assumptions, such as the hardness of the factoring problem, the hardness of the discrete logarithm problem, the hardness of DH problem and DDH assumption, etc.

2.3 Definition

We provide definitions for a key-evolving public-key encryption scheme. In the definition, we assume that there exists a secure device, which saves updating information of secret key for the decryptor.

Definition 3 A key-evolving public key encryption scheme KE-ENC consists

of four algorithms $\langle KG, UPD, E, D \rangle$:

1. *Key generation algorithm KG*: it is a probabilistic polynomial-time algorithm that takes as input a security parameter n and possibly other parameters and returns a base public key PK and corresponding base private key SK_0 . That is, $KG(1^n) = (PK, SK_0, s)$, where s is the system secret (trapdoor). If TA is involved, KG distributes s to it; otherwise, KG simply discards s .
2. *Private key update algorithm UPD*: it takes as input the public key PK , the private key SK_{j-1} of time period $j - 1, j \geq 1$, and outputs the new private key SK_j of time period j . That is, $UPD(PK, SK_{j-1}, j) = SK_j$. If TA is involved, UPD interacts with TA to compute SK_j .
3. *Encryption algorithm E*: it takes as input the base public key PK , a message m , and the time-period indicator j and outputs the ciphertext c of m at time period j . We use $E(PK, m, j) = \langle j, c \rangle$ to denote the encryption. E might be probabilistic.
4. *Decryption algorithm D*: it takes as input the ciphertext $\langle j, c \rangle$ of time period j and the private key $SK_{j'}$ of time period j' and outputs m if and only if $j = j'$. That is, $D(SK_{j'}, E(PK, m, j)) = m$.

We set no limit on the number of time periods. Key evolving can continue till the limit set by the security parameter n . The maximum number of time periods is 2^n .

We assume a single TA for simplicity. In practicality, we distribute trust to multiple trusted agents such that each TA_i holds a share s_i of the system secret s . The decryptor with private key SK_{j-1} and the TA's together can compute SK_j in a secure way, such as, through the secure multi-party computation.

Note that the time period is part of the ciphertext. When the decryptor gets $\langle j, c \rangle$, he uses his private key SK_j to decrypt the ciphertext. It may be that the decryptor gets $\langle j, c \rangle$ at time period $j', j' > j$. In this case, the decryptor

cannot decrypt it. Therefore, KE-ENC is better used for applications that need on-line decryption.

The key-evolving scheme cannot guarantee the desired security if the decryptor does not delete the old private key after getting the new one. It would be better to ensure erasure of old private keys by some system mechanism.

In addition, we consider the security of the scheme after the attacker gets some private keys additionally. Our schemes achieve the z -resilient property, that is, an adversary who gets z private keys does not obtain any information about a ciphertext.

Definition 4 (Resilience) *Assume a security model for public-key encryption scheme. A key-evolving public-key encryption scheme $\text{KE-ENC}=(KG, UPD, E, D)$ is z -resilient if the attacker cannot break the encryption scheme under the assumed security model even if he gets z private keys $SK_{j_1}, SK_{j_2}, \dots, SK_{j_z}$.*

Since the attacker gets z private keys, breaking does not include obtaining the corresponding plaintext of a ciphertext that is encrypted with any of the private keys.



Chapter 3

Key-Evolving Public Key Encryption Schemes

In the chapter, we present three key-evolving encryption schemes. The first KE-ENC scheme is based on the discrete logarithm problem, which is z -resilient and semantically secure against passive adversaries. We then modify the first scheme to become semantically secure against the adaptive chosen ciphertext attack under the random oracle model. Finally, we present the key-evolving encryption scheme under the standard model. Before describing the encryption schemes, we list notations used in this chapter in next section.

3.1 KE-ENC against passive adversaries

Our first scheme KEENC_{BASIC} is shown in Figure 3.1 and 3.2. The scheme consists of four algorithms: the key generation (KG) algorithm, the key update (UPD) algorithm, the encryption (E) algorithm and the decryption (D) algorithm. Let $f(x)$ denote a polynomial function with z degrees, says $f(x) = \sum_{i=0}^{i=z} a_i x^i$. We also assume that there are z TAs. The main idea is to treat $f(j)$ as the secret key of time period j . The public key of time period j , $g^{f(j)}$, is computed from PK and time j . Initially, a decryptor randomly selects a z -degree polynomial function. The decryptor keeps $f(x)$ secretly and

publishes the base public key $PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z} \rangle$. Then, the decryptor holds the secret key $f(0)$ and distributes $f(x_j)$ to TA_j for $1 \leq j \leq z$. When time transfers from $j - 1$ to j , the decryptor updates his secret key from SK_{j-1} to SK_j with the help of the TAs and deletes SK_{j-1} , immediately. We use the ElGamal-like [27] encryption scheme for encryption and decryption. When an encryptor wants to send a message to the decryptor at the time period j , the encryptor first computes PK_j from the base public key. Then, he computes ciphertext $c = \langle j', \alpha, s \rangle$ (refer to Figure 3.2) and sends c to the decryptor. The decryptor with SK_j can decrypt it if $j' = j$; otherwise, he discards the message. Note that $p = 2q + 1$, where p, q are primes. Then $G_q = QR_p = \{g^2 \pmod p | g \in G\}$. That is, all elements in G_q are quadratic residues. Since G_q 's order is prime, each element in G_q is a generator except 1. One cannot get one-bit information of quadratic residuosity.

Correctness. The correctness of decryption follows easily since

$$s/\alpha^{f(j)} = m \cdot \prod_{i=0}^z (g^{a_i})^{kj^i} / g^{kf(j)} = m \cdot g^{kf(j)} / g^{kf(j)} = m.$$

Efficiency. In each time period j , we can pre-compute

$$\prod_{i=0}^z (g^{a_i})^{j^i} = g^{a_0} (g^{a_1} (g^{a_2} (\dots)^j)^j)^j = g^{f(j)}.$$

It needs $z + 1$ modular exponentiations and z modular multiplications, which is independent of the time period j . After pre-computation, each encryption takes 2 modular exponentiations and 1 modular multiplication only. Each decryption takes one modular exponentiation and one modular division. Therefore, our scheme is very efficient in computation.

The public key consists of $(z + 1)$ n -bit values and the private key consists of one n -bit value only, which are both independent of the total number of time periods. The number of time periods can be as large as 2^n .

1. Algorithm $\text{KG}(1^n, z)$:

- (a) Randomly an n -bit prime $p = 2q + 1$, where q is also a prime. All operations work over Z_p except being stated otherwise. Let G_q be the subgroup of order q in Z_p^* and g be a generator of G_q .
- (b) Randomly select a degree- z polynomial $f(x) = \sum_{i=0}^z a_i x^i \bmod q$.
- (c) Set the public key and base private key as

$$PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z} \rangle \text{ and } SK_0 = \langle f(0) \rangle.$$

- (d) Let TA hold $f(x_j)$, for some random $x_j \in Z_q, 1 \leq j \leq z$.

2. Algorithm $\text{UPD}(PK, SK_{j-1})$: the decryptor Bob and TA together compute $SK_j = f(j)$ from their shares in a secure distributed way.



Figure 3.1: KEENCBASIC(part1) – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against passive adversaries.

3.1.1 Security analysis

We now show that KEENCBASIC is semantically secure against passive adversaries even the z decryption keys are disclosed. Let the z decryption keys be disclosed at time period j_1, \dots, j_z . Our scheme achieves forward and backward securities since an adversary cannot find two messages m_0 and m_1 whose ciphertexts are polynomially distinguishable at any time period j , $j \neq j_l, 1 \leq l \leq z$. That is, an adversary cannot get more information at any other time period, except time period j_1, \dots, j_z . This property guarantees our scheme with forward and backward securities.

Theorem 5 *Assume that the DDH problem is hard. For KEENCBASIC, given public key $PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z} \rangle$ and z private keys $SK_{j_1}, SK_{j_2}, \dots, SK_{j_z}$, no*

-
1. Algorithm $E(PK, m, j)$: randomly select $k \in Z_q$, $m \in G_q$ and compute

$$\alpha = g^k, \quad s = m \cdot \prod_{i=0}^z (g^{a_i})^{kj^i} = m \cdot g^{kf(j)}$$

and return the ciphertext $\langle j, \alpha, s \rangle$.

2. Algorithm $D(SK_j, \langle j, \alpha, s \rangle)$: compute and return $m = s/\alpha^{f(j)}$, where $SK_j = f(j)$.
-

Figure 3.2: KEENCBASIC(part2) – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against passive adversaries.

probabilistic polynomial-time adversary can distinguish the ciphertexts of any two messages m_0 and $m_1 \in G_q$ at time period j , $j \neq j_l, 1 \leq l \leq z$.

Proof. We reduce the DDH problem to the distinguishing problem of m_0 and m_1 at time period j . Assume that m_0 and m_1 are distinguishable with a non-negligible advantage $\epsilon(n)$ by a probabilistic polynomial-time adversary \mathcal{A} . We construct another probabilistic polynomial-time adversary \mathcal{B} to solve the DDH problem (with a non-negligible advantage $\epsilon(n)$).

Let $\langle g_1, g_2, u_1, u_2 \rangle$ be the input to the DDH problem. We let $a = \log_{g_1} g_2$ and randomly select $b_{j_1}, b_{j_2}, \dots, b_{j_z} \in Z_q$. There is a degree- z polynomial $f'(x) = \sum_{i=0}^z a'_i x^i \pmod q$ that passes $k + 1$ points $(j, a), (j_1, b_{j_1}), (j_2, b_{j_2}), \dots, (j_z, b_{j_z})$. Note that $a = f'(j)$. Although we don't know a , we can compute $g^{a'_i}$, $0 \leq i \leq z$, by Lagrange's interpolation method. We set the public key $PK = \langle g^{a'_0}, g^{a'_1}, \dots, g^{a'_z} \rangle$ with the base generator g_1 and z private keys $SK_{j_i} = \langle b_{j_i} \rangle$, $1 \leq i \leq z$. To pose a ciphertext challenge to \mathcal{A} , the encryption oracle randomly selects a bit b and computes the ciphertext $\langle j, u_1, m_b u_2 \rangle$ of m_b . The adversary \mathcal{B} outputs 1 for the input (g_1, g_2, u_1, u_2) if and only if \mathcal{A} 's output is equal to b for the challenge $(j, u_1, m_b u_2)$. We can see that

if $(g_1, g_2, u_1, u_2) = (g_1, g_2, g_1^r, g_2^r)$, $r \in Z_q$, the ciphertext outputted by the encryption oracle has the right distribution for each m_0 and m_1 . That is, $\Pr[\mathcal{A}(PK, \langle j, u_1, m_b u_2 \rangle) = b] = 1/2 + \epsilon(n)$. If $(g_1, g_2, u_1, u_2) = (g_1, g_2, g_1^{r_1}, g_2^{r_2})$, $r_1, r_2 \in Z_q$, the distributions of ciphertexts for m_0 and m_1 are equal. Thus, $\Pr[\mathcal{A}(PK, \langle j, u_1, m_b u_2 \rangle) = b] = 1/2$. Therefore, \mathcal{B} solves the DDH problem with a non-negligible advantage $\epsilon(n)$. \square

3.2 KE-ENC against adaptive chosen ciphertext attack

We propose two secure public key encryption schemes under the random oracle model and the standard model.

3.2.1 Under the random oracle model

We modify KEENCBASIC to be semantically secure against the adaptive chosen ciphertext attack under the random oracle model. The scheme also consists of four algorithms: KG, UPD, E and D. The modified scheme, KEENCROM, is shown in Figure 3.3. Let $f(x) = \sum_{i=0}^{i=z} a_i x^i$. We treat $f(j)$ as the secret key and $g^{f(j)}$ the public key of time period j . The KG and UPD are similar to these of the KEENCBASIC, but the public key contains three additional hash functions H_1, H_2 and H_3 in which H_1, H_2, H_3 are random oracle hash functions with output length dependent on n . The idea is to use randomness of hash functions. In encryption algorithm, we construct a (probabilistic) trapdoor one-way permutation

$$h_{j,y}(r) = (g^k, r \cdot y^k, k \oplus H(j, r))$$

for time period j , where $y = g^{f(j)}$ and the trapdoor is $f(j)$. The message is encrypted with one-time pad $H_2(j, r, k)$. The hash value $H_3(j, r, k, m)$ forces the querist to be aware of m . The ciphertext c is $\langle j, \alpha, \beta_1, \beta_2, s, h \rangle$ (see Figure 3.3

for detail). When the decryptor wants to decrypt the ciphertext c , he checks whether $\alpha = g^k$ and $h = H_3(j, r, k, m)$ in which r , k and m refers to Figure 3.3. The lengths of the three hash functions are not the same. The length of H_1 is the same to $|k|$ where $|k|$ denotes the bit-length of k . The length of H_2 is the same to $|m|$ where $|m|$ deontes the bit-length of m .

Correctness. We can see that

$$\begin{aligned}\beta_1/\alpha^{f(j)} &= r \cdot g^{kf(j)}/g^{kf(j)} = r, \\ \beta_2 \oplus H_1(j, r) &= (k \oplus H_1(j, r)) \oplus H_1(j, r) = k, \text{ and} \\ s \oplus H_2(j, r, k) &= (m \oplus H_2(j, r, k)) \oplus H_2(j, r, k) = m.\end{aligned}$$

Efficiency. Encryption and decryption computation time is similar to that of KEENCBASIC. After pre-computation, each encryption takes 2 modular exponentiations, 1 modular multiplication and 3 hash operations. Each decryption takes 1 modular exponentiation, 1 modular division and 3 hash operations. Again, computation time is independent of the time period j . Therefore, this scheme is as efficient as KEENCBASIC.

Again, KEENCROM's public key consists of $(z+1)$ n -bit values and private key consists of only one n -bit value, which are both independent of the total number of time periods.

3.2.2 Security analysis

Assume the random oracle model, which postulates that H_1 , H_2 and H_3 are "truly random" hash functions. We show that KEENCROM is semantically secure against the adaptive chosen ciphertext attack. As KEENCBASIC stated, assume that an adversary can get z decryption keys in advance. Let the z decryption keys be disclosed at time period j_1, \dots, j_z . Since an adversary cannot find two messages m_0 and m_1 whose ciphertexts are polynomially distinguishable at any time period j , $j \neq j_l, 1 \leq l \leq z$ such that our scheme achieves forward and backward securities.

Theorem 6 *Assume the random oracle model and that the discrete logarithm problem is hard. For KEENCROM, given public key $PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z} \rangle$ and z private keys $SK_{j_1}, SK_{j_2}, \dots, SK_{j_z}$, no probabilistic polynomial-time adversary with access to the decryption oracle is able to find m_0 and m_1 in G_q such that their ciphertexts at time period j , $j \neq j_l, 1 \leq l \leq z$, are distinguishable.*

Proof. Let n be the security parameter (or complexity measure). Let an adversary \mathcal{A} that can distinguish the ciphertexts of two messages m_0 and m_1 with a non-negligible probability even knowing z private keys and being able to query the decryption oracle adaptively. \mathcal{A} consists of two probabilistic polynomial-time procedures A_1 and A_2 . A_1 takes as input the public key PK and z private keys, makes some queries to the decryption and hash oracles, and outputs two messages m_0 and m_1 . A_2 takes as input the public key PK , z private keys, m_0 , m_1 , and the ciphertext $E(PK, m_b, j)$, queries the decryption and hash oracles adaptively, and outputs b' , where b is a random bit. We say that A attacks the scheme successfully if $\Pr[b = b'] = 1/2 + \epsilon(n)$ for some non-negligible function $\epsilon(n)$, where the probability is taken over b and the internal coin tosses of A_1 , A_2 , KG and z private keys. We show that we can construct a probabilistic polynomial-time algorithm \mathcal{B} to solve the discrete logarithm problem with a non-negligible probability by using \mathcal{A} .

By the random oracle paradigm, one cannot get hash results without querying the hash oracles H_1 , H_2 and H_3 except with a negligible probability. It means that one has to know the input to the hash oracles. Therefore, one has to know r and k , and thus can solve the discrete logarithm problem, which contradicts to the assumption.

Assume that \mathcal{A} makes q_d queries to the decryption oracle and q_h queries to the hash oracles within time bound $t(n)$ and gains an $\epsilon(n)$ advantage. Without loss of generality, let the hash results of H_1 , H_2 and H_3 be of length n . Our simulation for the hash and decryption oracles is as follows. The simulator S maintains a table T for the random oracles for consistency. When \mathcal{A} makes a

hash query (j', r') to H_1 , S checks whether (j', r') is already in T . If (j', r') is in T , it returns the hash result in T ; otherwise, it returns a random value c' and puts $\langle (j', r'), c' \rangle$ into T . When \mathcal{A} makes a hash query (j', r', k') to H_2 , S checks whether the query is in T . If yes, S returns the hash result t' to \mathcal{A} ; otherwise, he randomly selects t' and adds $\langle (j', r', k'), t' \rangle$ to T and returns t' to \mathcal{A} . The same is done for the queries to H_3 . If (j', r', k', m') is queried in T , S returns the hash result h' to \mathcal{A} . Otherwise, S selects a random value h' , adds $\langle (j', r', k', m'), h' \rangle$ to T , and returns h' to \mathcal{A} . For the decryption oracle, if $(j', \alpha', \beta'_1, \beta'_2, s', h')$ is queried, S checks whether (j', r') , (j', r', k') , and (j', r', k', m') with $\alpha' = g^{k'}$ have been asked. If either one of them is not asked, S returns "invalid ciphertext". Otherwise, we can verify $(j', \alpha', \beta'_1, \beta'_2, s', h')$ by checking $\alpha' = g^{k'}$, $\beta'_1 = r' \cdot (g^{f(j')})^{k'}$, $\beta'_2 = k' \oplus c_1$, $s' = m' \oplus c_2$ and $h' = c_3$ where $c_1 = h_1(j', r')$, $c_2 = h_2(j', r', k')$ and $c_3 = h_3(j', r', k', m')$ are in T and m' is the corresponding plaintext. If it is so, it returns m' ; otherwise, it returns '?', which means that the input ciphertext is invalid. Note that S may be wrong on returning '?' since $(j', \alpha', \beta'_1, \beta'_2, s', h')$ may be valid. But, this occurs with probability $1/2^{3n}$ only due to the random oracle model. Now, S constructs a ciphertext $E(PK, m_b, j) = (j', \alpha', \beta'_1, \beta'_2, s', h')$ such that $s' = m_b \oplus h_2(j', r', k')$ and $b = 0$, or 1, then S runs $A_2^{H_1, H_2, H_3, D^{H_1, H_2, H_3}}(PK, SK's, m_0, m_1, E(PK, m_b, j))$ as it did to A_1 .

Let E_1 be the event that \mathcal{A} does make a query (j, r) to the H_1 hash oracle, or a query (j, r, k) to the H_2 hash oracle, or a query (j, r, k, m) to the H_3 hash oracle. Let E_2 be the event that \mathcal{A} makes a correct decryption query $(j', \alpha', \beta'_1, \beta'_2, s', h')$, but without querying appropriate hash queries. We have

$$\Pr[E_2] \leq q_d 2^{-3n} \leq t(n) 2^{-3n}.$$

Since without querying the hash oracles on proper r , k and m , \mathcal{A} has no advantage in distinguishing m_0 from m_1 . Thus, we have

$$\Pr[A_2(PK, SK's, m_0, m_1, E(PK, m_b, j)) = b | \neg(E_1 \vee E_2)] = 1/2.$$

Since \mathcal{A} has $\epsilon(n)$ advantage in guessing b , we have

$$\begin{aligned}
1/2 + \epsilon(n) &= \Pr[A_2(PK, SK's, m_0, m_1, E(PK, m_b, j)) = b] \\
&= \Pr[A_2(\cdot) = b|E_2] \Pr[E_2] \\
&\quad + \Pr[A_2(\cdot) = b|\bar{E}_2 \wedge E_1] \cdot \Pr[E_1 \wedge \bar{E}_2] \\
&\quad + \Pr[A_2(\cdot) = b|\bar{E}_2 \wedge \bar{E}_1] \cdot \Pr[\bar{E}_1 \wedge \bar{E}_2] \\
&\leq t(n)2^{-3n} + \Pr[E_1] + (1/2) \Pr[\bar{E}_1].
\end{aligned}$$

This implies that $\Pr[E_1] \geq 2\epsilon(n) - t(n)2^{-3n+1}$.

The algorithm \mathcal{B} of solving the discrete logarithm problem works as follows. On input (g, y) , it randomly selects a degree- z polynomial $f'(x) = \sum_{i=0}^z a_i x^i \bmod q$. It sets the decryption key SK_j of time period j as $f'(j)$ and $SK_{j_i} = f'(j_i)$ as the given exposed decryption key of time periods j_i , $1 \leq i \leq z$. \mathcal{B} calls $A_1(g^{a_0}, g^{a_1}, \dots, g^{a_z}, SK_{j_1}, SK_{j_2}, \dots, SK_{j_z})$ to find m_0 and m_1 with the simulator S to answer queries. It then randomly selects r , c_1 , c_2 and c_3 and feeds the ciphertext $\langle j, c \rangle = \langle j, y, ry^{SK_j}, c_1, c_2, c_3 \rangle$ to A_2 , where c_1 , c_2 and c_3 are randomly chosen and consistent with S 's answers to hash queries. Finally, \mathcal{B} outputs k if (j, r) , (j, r, k) , or (j, r, k, m) have been queried to the hash oracles and $g^k = y$, where k is either queried in (j, r, k) or (j, r, k, m) , or is equal to $c_1 \oplus H_1(j, r)$.

By the above argument, A_2 queries (j, r) , (j, r, k) or (j, r, k, m) of appropriate form with probability $\Pr[E_1] = 2\epsilon(n) - t(n)2^{3n+1}$. Therefore, the probability of $g^k = y$ is $2\epsilon(n) - t(n)2^{-3n+1}$ at least, which is non-negligible. \square

3.2.3 Under the standard model

Now, we modify KEENC BASIC to achieve the security against the adaptive chosen ciphertext attack under the standard model. The scheme is shown in Figure 3.4 and Figure 3.5. Comparison to KEENC ROM, we assume that a hash function H is collision-resistant but not random. For such a function H , it is infeasible to find two distinct inputs v_1 and v_2 such that $H(v_1) = H(v_2)$. A hash function H is chosen from the collision-resistant hash family \mathcal{H} .

At first, we choose five z -degree polynomial functions at random, $f_{a_0}(x), f_{a_1}(x), f_{b_1}(x), f_{a_2}(x), f_{b_2}(x)$, where $f_{a_j}(x) = \sum_{i=0}^z a_{j,i}x^i$ for $0 \leq j \leq 2$ and $f_{b_j}(x) = \sum_{i=0}^z b_{j,i}x^i$ for $1 \leq j \leq 2$. The decryptor saves the constant term of each of polynomial functions as the initial secret key SK_0 . The decryptor and TA compute the secret key SK_j of time period j in a secure way. The public key consists of g, h, q, p, H and commitments of the coefficients of these polynomials.

To encrypt a message m at time period j , the encryptor computes the following value $\langle j, \alpha, \beta, s, \delta \rangle$, where $\alpha = g^k$, $\beta = h^k$, $s = m \cdot g^{kf_{a_0}(j)}$, $\delta = g^{kf_{a_1}(j)}h^{kf_{b_1}(j)}(g^{kf_{a_2}(j)}h^{kf_{b_2}(j)})^v$ and $v = H(j, \alpha, \beta, s)$ for a random $k \in Z_q$. The decryptor can decrypt as long as he has $SK_j = \{f_{a_0}(j), f_{a_1}(j), f_{b_1}(j), f_{a_2}(j), f_{b_2}(j)\}$. The decryptor computes $v' = H(j, \alpha, \beta, s)$ and checks if $\alpha^{f_{a_1}(j)+f_{a_2}(j)v'}\beta^{f_{b_1}(j)+f_{b_2}(j)v'} = \delta$. If so, he decrypts and obtains m .

Correctness. Let $SK_j = \{f_{a_0}(j), f_{a_1}(j), f_{b_1}(j), f_{a_2}(j), f_{b_2}(j)\}$ and a ciphertext of m is $\langle j, \alpha, \beta, s, \delta \rangle$. If $v = H(j, \alpha, \beta, s)$, then $\alpha^{f_{a_1}(j)+f_{a_2}(j)v}\beta^{f_{b_1}(j)+f_{b_2}(j)v} = g^{kf_{a_1}(j)}g^{kf_{a_2}(j)v}h^{kf_{b_1}(j)}h^{kf_{b_2}(j)v} = g^{kf_{a_1}(j)}h^{kf_{b_1}(j)}(g^{kf_{a_2}(j)v}h^{kf_{b_2}(j)v}) = \delta$. We can obtain the message m by computing $s/\alpha^{f_{a_0}(j)} = m \cdot g^{kf_{a_0}(j)}/\alpha^{f_{a_0}(j)} = m$.

Efficiency. After pre-computation, each encryption takes 5 modular exponentiations, 2 modular multiplications and 1 hash operations. Each decryption takes 3 modular exponentiations, 1 modular division, 2 modular multiplications and 1 hash operations. Computation time is independent of the time period j . Therefore, this scheme is as efficient as KEENCBASIC.

Again, KEENCSTM's public key consists of $3(z+1)$ n -bit values and private key consists of only one $5n$ -bit value, which are both independent of the total number of time periods.

3.2.4 Security analysis

Assume the DDH assumption. We show that KEENCSTM is semantically secure against the adaptive chosen ciphertext attack. For time period j , the

decryption key SK_j doesn't be disclosed, an adversary cannot distinguish $E(PK, m_0, j)$ from $E(PK, m_1, j)$. Therefore, our scheme has forward and backward securities.

Theorem 7 *Assume the DDH problem is hard. For KEENCSTM, given public key $PK = \langle g, h, q, H, \{w_i, c_i, d_i\}_{0 \leq i \leq z} \rangle$ and z private keys $SK_{j_i}^*, 1 \leq i \leq z$, no probabilistic polynomial-time adversary with access to the decryption oracle is able to find m_0 and m_1 in G_q such that their ciphertexts at time period j , $j \neq j_l, 1 \leq l \leq z$, are distinguishable.*

Proof. Assume that the scheme does not have the desired properties. There is an adversary \mathcal{A} that can distinguish the ciphertexts of the two messages m_0 and m_1 with a non-negligible probability even knowing z private keys and being able to query the decryption oracle (\mathcal{DO}) adaptively. \mathcal{A} has two probabilistic polynomial-time procedures A_1 and A_2 . A_1 takes as input the public key and z private keys, makes some chosen-ciphertext queries to (\mathcal{DO}), and outputs two messages m_0 and m_1 . A_2 takes as input the public key PK , z private keys, m_0 , m_1 , and the ciphertext $E(PK, m_b, j)$, queries (\mathcal{DO}) adaptively, and outputs b' , where b is a random bit. We say that \mathcal{A} attacks the scheme successfully if $\Pr[b = b'] = 1/2 + \epsilon(n)$ for some non-negligible function $\epsilon(n)$, where the probability is taken over b and the internal coin tosses of A_1 , A_2 , KG and z private keys. We show that we can use \mathcal{A} to construct a probabilistic polynomial-time algorithm \mathcal{B} for solving the DDH problem with a non-negligible probability.

Let the input of DDH be $\langle g_1, g_2, u_1, u_2 \rangle$. We construct a simulator \mathcal{S} that simulates \mathcal{A} 's view in its attack on the algorithm. The simulator \mathcal{S} contains an encryption oracle \mathcal{EO} and a decryption oracle \mathcal{DO} . We will show that the simulation of \mathcal{A} 's view will be nearly perfect if the quadruple is from D and \mathcal{A} 's advantage is negligible if the quadruple is from R . We construct the simulator \mathcal{S} as follows.

1. **Key setup.** The public key is constructed as follows.

- (a) Randomly select $a_{j_1}, \dots, a_{j_z}, b_{j_1}, \dots, b_{j_z}$ over Z_q .
- (b) There is a z -degree polynomial function $f''_{a_0}(x) = \sum_{i=0}^z a''_{0,i} x^i \pmod q$ that passes $z+1$ points $(j, a), (j_1, a_{j_1}), \dots, (j_z, a_{j_z})$, where $a = \log_g h$ is unknown. Randomly select $z+1$ points $(j, e \cdot a), (j_1, e \cdot b_{j_1}), \dots, (j_z, e \cdot b_{j_z})$ and let $f''_{b_0}(x) = \sum_{i=0}^z b''_{0,i} x^i \pmod q$ passes these points. Let $w'_i = g^{a''_{0,i}} h^{b''_{0,i}}$ for $0 \leq i \leq z$. Although a is unknown, we can compute w'_i for $0 \leq i \leq z$ by Lagrange interpolation. Thus $f'_{a_0}(x) = \sum_{i=0}^z (a''_{0,i} + ab''_{0,i}) x^i \pmod q$. Now, we construct c'_i and d'_i as follows. Let $f'_{a_1}(x) = \sum_{i=0}^z a'_{1,i} x^i \pmod q$ that passes $z+1$ points $(j, a + r_{1,1}), (j_1, r_{1,2}), \dots, (j_z, r_{1,z+1})$, $f'_{a_2}(x) = \sum_{i=0}^z a'_{2,i} x^i \pmod q$ that passes $z+1$ points $(j, a + r_{2,1}), (j_1, r_{2,2}), \dots, (j_z, r_{2,z+1})$, $f'_{b_1}(x) = \sum_{i=0}^z b'_{1,i} x^i \pmod q$ that passes $z+1$ points $(j, a + r_{3,1}), (j_1, r_{3,2}), \dots, (j_z, r_{3,z+1})$ and $f'_{b_2}(x) = \sum_{i=0}^z b'_{2,i} x^i \pmod q$ that passes $z+1$ points $(j, a + r_{4,1}), (j_1, r_{4,2}), \dots, (j_z, r_{4,z+1})$, where $r_{i,j} \in R$ for $1 \leq i \leq 4, 1 \leq j \leq z+1$. Then, $c'_i = g^{a'_{1,i}} h^{b'_{1,i}}$ and $d'_i = g^{a'_{2,i}} h^{b'_{2,i}}$ for $0 \leq i \leq z$ by Lagrange interpolation. We can fix $e = 0$.
- (c) Then we set the public key as follows.

$$PK = \langle g, h, q, H, \{w'_i, c'_i, d'_i\}_{0 \leq i \leq z} \rangle,$$

where H is a family of collision-resistant hash functions, $g = g_1$ and $h = g_2$.

The key generation is a bit different from the actual cryptosystem; however, the effect is the same.

2. **Challenge.** Feed the public key PK and the z private keys $SK'_j = \{b_{j_{i-1}}, eb_{j_{i-1}}, r_{1,i}, r_{2,i}, r_{3,i}, r_{4,i}\}, 2 \leq i \leq z+1$, to A_1 and get two messages m_0 and m_1 in G_q .
3. **Encryption.** Randomly select a bit $b \in_R \{0, 1\}$ and compute

$$\alpha' = u_1, \beta' = u_2, \quad s' = m_b \cdot u_1^{f''_{a_0}(j)} u_2^{f''_{b_0}(j)}, \text{ and}$$

$$\delta' = u_1^{f'_{a_1}(j)} u_2^{f'_{b_1}(j)} (u_1^{f'_{a_2}(j)} u_2^{f'_{b_2}(j)})^{v'},$$

where $v' = H(j, \alpha', \beta', s')$. Then, the encryptor returns the ciphertext $c = \langle j, \alpha', \beta', s', \delta' \rangle$.

4. **Decryption.** Given the ciphertext c , \mathcal{DO} checks if c is valid by verifying

$$u_1^{f'_{a_1}(j)+f'_{a_2}(j)v'} u_2^{f'_{b_1}(j)+f'_{b_2}(j)v'} = \delta'.$$

If it is not valid, the oracle rejects it. Otherwise, \mathcal{DO} returns

$$m = s' / u_1^{f''_{a_0}(j)} u_2^{f''_{b_0}(j)}.$$

This completes the description of \mathcal{S} . The adversary \mathcal{B} takes as input (g_1, g_2, u_1, u_2) and outputs 1 if $b = A_2(s')$. To complete the proof, we show:

1. If (g_1, g_2, u_1, u_2) is from D , the joint distribution of the adversary's view and the hidden bit b is statically indistinguishable from that in the actual attack.
2. If (g_1, g_2, u_1, u_2) comes from R , the distribution of the hidden bit b is independent of the adversary's view.

The theorem follows immediately from the following two lemmas.

Lemma 8 *If the S 's input (g_1, g_2, u_1, u_2) is from D , the joint distribution of the adversary's view and the hidden bit b is statically indistinguishable from that in the actual attack.*

We need argue two things. One is that the output of \mathcal{DO} and \mathcal{EO} has the right distribution. The other is that \mathcal{DO} rejects all invalid ciphertexts except with a negligible probability. Since the input comes from D , we have that $u_1 = g_1^r$ and $u_2 = g_2^r$ for some r . Thus, $\alpha' = u_1$, $\beta' = u_2$, $s' = m_b \cdot u_1^{f''_{a_0}(j)} u_2^{f''_{b_0}(j)}$ and $\delta' = u_1^{f'_{a_1}(j)} u_2^{f'_{b_1}(j)} (u_1^{f'_{a_2}(j)} u_2^{f'_{b_2}(j)})^{v'}$, where s' and δ' can be computed by the Lagrange interpolation and $v' = H(j, \alpha', \beta', s')$. Hence, the output of

the encryption oracle \mathcal{EO} has the right distribution. In addition, the ciphertext $c = \langle j, \alpha', \beta', s', \delta' \rangle$ is a valid ciphertext. Therefore, \mathcal{DO} outputs $m_b = s' / u_1^{f''_{a_0}(j)} u_2^{f''_{b_0}(j)}$.

Moreover, we will show that \mathcal{DO} rejects all invalid ciphertexts, except with a negligible probability. Consider that the invalid ciphertext $\langle j, \alpha', \beta', s', \delta' \rangle$. Since the adversary knows at most z values of $f''_{a_0}(\cdot)$, $f''_{b_0}(\cdot)$, $f'_{a_1}(\cdot)$, $f'_{b_1}(\cdot)$, $f'_{a_2}(\cdot)$ and $f'_{b_2}(\cdot)$ other than j . Let \log denote \log_g , $\log h = a$, $c_j = g^{f'_{a_1}(j)} h^{f'_{b_1}(j)}$ and $d_j = g^{f'_{a_2}(j)} h^{f'_{b_2}(j)}$. From the public key c_j and d_j , we obtain two equations:

$$\log c_j = f'_{a_1}(j) + a f'_{b_1}(j), \quad (3.1)$$

$$\log d_j = f'_{a_2}(j) + a f'_{b_2}(j). \quad (3.2)$$

From the output of the \mathcal{EO} , we obtain another equation:

$$\log \delta' = r \cdot f'_{a_1}(j) + ra \cdot f'_{b_1}(j) + v' r \cdot f'_{a_2}(j) + v' ra \cdot f'_{b_2}(j). \quad (3.3)$$

If the adversary submits an invalid ciphertext $(j, \alpha^*, \beta^*, s^*, \delta^*)$ to \mathcal{DO} , ie., $r_1 = \log u_1 \neq \log_{g_2} u_2 = r_2$. \mathcal{DO} will reject, unless the point $\mathbf{P} (f'_{a_1}(j), f'_{a_2}(j), f'_{b_1}(j), f'_{b_2}(j))$ lies on the hyperplane \mathbf{H} defined by

$$\log \delta^* = r_1 \cdot f'_{a_1}(j) + r_2 a \cdot f'_{b_1}(j) + v^* r_1 \cdot f'_{a_2}(j) + v^* r_2 a \cdot f'_{b_2}(j), \quad (3.4)$$

where $v^* = H(j, \alpha^*, \beta^*, s^*)$. Since Equations 3.1, 3.2 and 3.3 are linearly independent, the hyperplane \mathbf{H} intersects \mathbf{P} at a line.

The first time the \mathcal{DO} rejects an invalid ciphertext, except with probability $1/q$. However, the i_{th} query will be rejected, except with the probability at least $1/(q - i + 1)$. Following the result above, \mathcal{DO} will reject all invalid ciphertext, except with negligible probability.

Lemma 9 *If the S 's input (g_1, g_2, u_1, u_2) is from R , the distribution of the hidden bit b is independent of the adversary's view.*

We should show that if \mathcal{DO} rejects all invalid ciphertexts, the distribution of the hidden bit b is independent of the adversary's view. Let $r_1 = \log u_1$

and $r_2 = \log_{g_2} u_2$. Assume that $r_1 \neq r_2$. The public key $w_j = g^{f''_{a_0}(j)} h^{f''_{b_0}(j)}$ determines the equation:

$$\log w_j = f''_{a_0}(j) + a f''_{b_0}(j). \quad (3.5)$$

Moreover, if the \mathcal{DO} only decrypts valid ciphertexts $(\langle j, \alpha', \beta', s', \delta' \rangle)$, then the adversary obtains only linearly dependent relation $r' \log w_j = r' f''_{a_0}(j) + r' a f''_{b_0}(j)$, where $r' = \log u_1$. Thus, no further information about $(f''_{a_0}(j), f''_{b_0}(j))$ is leaked.

Consider that the output of \mathcal{EO} , we have $s' = m_b \cdot u_1^{f''_{a_0}(j)} u_2^{f''_{b_0}(j)}$. Let $\tau = u_1^{f''_{a_0}(j)} u_2^{f''_{b_0}(j)}$

$$\log \tau = r_1 f''_{a_0}(j) + r_2 a f''_{b_0}(j). \quad (3.6)$$

Equations 3.5 and 3.6 are linearly independent. We can view τ as a one-time pad. As a result the bit b is independent of the adversary's view.

Next, we prove that \mathcal{DO} will reject all invalid ciphertexts, except with a negligible probability. Based on the adversary's view, we examine the distribution of $\mathbf{P} = (f'_{a_1}(j), f'_{a_2}(j), f'_{b_1}(j), f'_{b_2}(j)) \in \mathbf{Z}_q^4$. From the output of \mathcal{EO} , we get the following equation:

$$\log \delta' = r_1 \cdot f'_{a_1}(j) + r_2 a \cdot f'_{b_1}(j) + v' r_1 \cdot f'_{a_2}(j) + v' r_2 a \cdot f'_{b_2}(j). \quad (3.7)$$

From the adversary's view, \mathbf{P} is a random point on the line \mathbf{L} formed by intersecting the hyperplane of Equations 3.1, 3.2 and 3.7. Let $r'_1 = \log u'_1$ and $r'_2 = \log_{g_2} u'_2$. If the submitted ciphertext $(\langle j, \alpha', \beta', s', \delta' \rangle)$ is invalid, there are three cases to consider:

1. **Case I.** $(\alpha', \beta', s') = (\alpha, \beta, s)$. Since $\delta' \neq \delta$, the decryption oracle still rejects.
2. **Case II.** $(\alpha', \beta', s') \neq (\alpha, \beta, s)$ but the hash value is the same. This immediately violates the collision-resistance of our hash function. Therefore, this cannot occur with non-negligible probability.

3. **Case III.** $(\alpha', \beta', s') \neq (\alpha, \beta, s)$ and the hash value is not the same. Unless the point \mathbf{P} satisfies the hyperplane $\log \delta'$. Otherwise, \mathcal{DO} will reject. Moreover, Equations 3.1, 3.2, 3.4 and 3.7 are linearly independent when $\alpha' \neq \alpha^*$, $r_1 \neq r'_1$ and $r_2 \neq r'_2$. It follows that the decryption oracle rejects, except with a negligible probability.

Now, we consider the case that when $t \neq j$, the tuple $(f'_{a_1}(j), f'_{a_2}(j), f'_{b_1}(j), f'_{b_2}(j), f'_{a_1}(t), f'_{a_2}(t), f'_{b_1}(t), f'_{b_2}(t))$ is uniformly distributed subject to several constraints. At first, we have Equations 3.1, 3.2 and 3.7. Next, we have the two equations from the public key: From the public key c_t and d_t , we obtain two equations:

$$\log c_t = f'_{a_1}(t) + a f'_{b_1}(t), \quad (3.8)$$

$$\log d_t = f'_{a_2}(t) + a f'_{b_2}(t). \quad (3.9)$$

Since the adversary could have z secret keys other than j and t , he can know z values of each of $f'_{a_1}(\cdot)$, $f'_{b_1}(\cdot)$, $f'_{a_2}(\cdot)$ and $f'_{b_2}(\cdot)$. Therefore, the following relations hold:

$$f'_{a_1}(j) + \lambda f'_{a_1}(t) = s_1 \quad (3.10)$$

$$f'_{a_2}(j) + \lambda f'_{a_2}(t) = s_2 \quad (3.11)$$

$$f'_{b_1}(j) + \lambda f'_{b_1}(t) = s_3 \quad (3.12)$$

$$f'_{b_2}(j) + \lambda f'_{b_2}(t) = s_4 \quad (3.13)$$

where λ is the Lagrange coefficient $\lambda = (j - j_1)(j - j_2) \cdots (j - j_z) / (t - j_1)(t - j_2) \cdots (t - j_z)$. Now we have more equations than unknowns. However, it is easy to see that Equation 3.12 is linearly dependent on Equations 3.1, 3.2 and 3.10 while Equation 3.13 is linearly dependent on Equations 3.8, 3.9 and 3.11. Thus, there are 7 linearly independent equations and 8 unknowns. Consider that the ciphertext $(\langle t, \alpha'', \beta'', s'', \delta'' \rangle)$ submitted by the adversary is invalid. \mathcal{DO} will reject unless

$$\log \delta'' = r_1 \cdot f'_{a_1}(t) + r_2 a \cdot f'_{b_1}(t) + v'' r_1 \cdot f'_{a_2}(t) + v'' r_2 a \cdot f'_{b_2}(t), \quad (3.14)$$

Looking at all 8 Equations 3.1, 3.2, 3.7- 3.11 and 3.14, we see that they are linearly independent when the following three conditions hold:

1. $\log u_1 \neq \log_{g_2} u_2$. This is true since (g_1, g_2, u_1, u_2) is a random tuple.
2. $\log \alpha'' \neq \log_{g_2} \beta''$. This is true since the ciphertext $(\langle t, \alpha'', \beta'', s'', \delta'' \rangle)$ is invalid.
3. $H(j, \alpha', \beta', s') \neq H(t, \alpha'', \beta'', s'')$. This is true since $j \neq t$ and H is chosen from a family of collision resistant functions. The collision resistance of H is necessary since the adversary's choice of j, t is not known in advance.

Therefore, the ciphertext is rejected except with negligible probability at most $1/q$. The i th query except with probability at most $1/(q-i+1)$. This completes the proof. □



1. Algorithm $\text{KG}(1^n, z)$:

- (a) Randomly select an n -bit prime $p = 2q + 1$, where q is also a prime.
Let G_q be the subgroup of order q in Z_p^* and g be a generator of G_q .
- (b) Randomly select a degree- z polynomial $f(x) = \sum_{i=0}^z a_i x^i \pmod{q}$.
- (c) H_1, H_2, H_3 are random oracle hash functions.
- (d) Set the public key and base private key as

$$PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z}, p, H_1, H_2, H_3 \rangle \text{ and } SK_0 = \langle f(0) \rangle.$$

- (e) Let TA hold $f(x_j), x_j \in Z_q, 1 \leq j \leq z$.

2. Algorithm $\text{UPD}(PK, SK_{j-1})$: the decryptor Bob and TA together compute $SK_j = \langle f(j) \rangle$ from their shares in a secure distributed way.

3. Algorithm $\text{E}^{H_1, H_2, H_3}(PK, m, j)$: randomly select $k \in Z_q$ and $r \in G_q$, compute

$$\alpha = g^k, \beta_1 = r \cdot \left(\prod_{i=0}^z (g^{a_i})^{j^i} \right)^k = r \cdot g^{f(j) \cdot k},$$

$$\beta_2 = k \oplus H_1(j, r), s = m \oplus H_2(j, r, k), h = H_3(j, r, k, m),$$

and return the ciphertext $\langle j, \alpha, \beta_1, \beta_2, s, h \rangle$.

4. Algorithm $\text{D}^{H_1, H_2, H_3}(SK_j, \langle j, \alpha, \beta_1, \beta_2, s, h \rangle)$:

- (a) Compute $r = \beta_1 / \alpha^{f(j)}, k = \beta_2 \oplus H_1(j, r)$ and $m = s \oplus H_2(j, r, k)$.
 - (b) Check whether $\alpha = g^k$ and $h = H_3(j, r, k, m)$. If it is so, return m ; otherwise, return '??'.
-

Figure 3.3: KEENCROM – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against the adaptive chosen ciphertext attack under the random oracle model.

1. Algorithm $\text{KG}(1^n, z)$: let H be a collision-resistant hash function selected from a family of collision-resistant hash functions.

(a) Randomly select an n -bit prime $p = 2q + 1$, where q is also a prime. All operations work over Z_p except being stated otherwise. Let G_q be the subgroup of order q in Z_p^* and g be a generator of G_q .

(b) Randomly select five degree- z polynomial functions $f_{a_j}(x) = \sum_{i=0}^z a_{j,i}x^i$ for $0 \leq j \leq 2$ and $f_{b_j}(x) = \sum_{i=0}^z b_{j,i}x^i$ for $1 \leq j \leq 2$.

(c) Set $w_i^* = g^{a_{0,i}}$, $c_i^* = g^{a_{1,i}}h^{b_{1,i}}$ and $d_i^* = g^{a_{2,i}}h^{b_{2,i}}$ for $0 \leq i \leq z$.

(d) Set the public key and base private key as

$$PK = \langle g, h, q, H, \{w_i^*, c_i^*, d_i^*\}_{0 \leq i \leq z} \rangle$$

and

$$SK_0 = \{f_{a_0}(0), f_{a_1}(0), f_{b_1}(0), f_{a_2}(0), f_{b_2}(0)\}.$$

(e) Let TA hold $\{f_{a_0}(x_j), f_{a_1}(x_j), f_{b_1}(x_j), f_{a_2}(x_j), f_{b_2}(x_j)\}$, $x_j \in Z_q$, $1 \leq j \leq z$.

Figure 3.4: KEENCSTM (part 1)– discrete logarithm based key-evolving scheme with z -resilience and semantic security against the adaptive chosen ciphertext attack under the standard model.

1. Algorithm $\text{UPD}(PK, SK_{j-1})$: the decryptor Bob and TA together compute $SK_j = \langle f_{a_0}(j), f_{a_1}(j), f_{b_1}(j), f_{a_2}(j), f_{b_2}(j) \rangle$ from their shares in a secure distributed way.

2. Algorithm $\text{E}(PK, m, j)$: randomly select $k \in Z_q$, compute

$$\alpha = g^k, \beta = h^k, \quad s = m \cdot \prod_{i=0}^z (w_i^*)^{kj^i} = m \cdot g^{kf_{a_0}(j)}, \text{ and}$$

$$\delta = \prod_{i=0}^z (c_i^*)^{kj^i} \cdot \left(\prod_{i=0}^z (d_i^*)^{kj^i} \right)^v = g^{kf_{a_1}(j)} h^{kf_{b_1}(j)} (g^{kf_{a_2}(j)} h^{kf_{b_2}(j)})^v,$$

where $v \stackrel{\text{def}}{=} H(j, \alpha, \beta, s)$. Then, the encryptor returns the ciphertext $C = \langle j, \alpha, \beta, s, \delta \rangle$.

3. Algorithm $\text{D}(SK_j^*, \langle j, \alpha, \beta, s, \delta \rangle)$.

- (a) Compute $v = H(j, \alpha, \beta, s)$.
 - (b) Check if $\alpha^{f_{a_1}(j)+f_{a_2}(j)v} \beta^{f_{b_1}(j)+f_{b_2}(j)v} = \delta$.
 - (c) If so, compute and return $m = s / \alpha^{f_{a_0}(j)}$.
-

Figure 3.5: KEENCSTM (part 2) – discrete logarithm based key-evolving encryption scheme with z -resilience and semantic security against the adaptive chosen ciphertext attack under the standard model.

Chapter 4

Distributed and Proactive Key-Evolving Encryption

In this chapter, we describe the procedure for a key evolving with TA. Then, we use the proactive mechanism to protect the TAs. Furthermore, we propose a key-evolving encryption in a distributed way. Multiple decryptors decrypts encrypted messages via distributed computing. Finally, we present how the proactive mechanism protects the secret of the decryptors. We assume that involved n parties are connected by a broadcast channel and any two parties are connected by a private channel such that a third party cannot get messages sent over the private channel.

4.1 Key evolving with TA

We propose two protocols for key evolving with TA. One is for distributing TA's secret. The other is for proactivizing TA's shares.

4.1.1 Distributing TA's secret

Assume that there exist z TA's and each TA_i holds a share $f(x_i)$, $1 \leq i \leq z$, where x_i 's are distinct and large enough so that the maximum time period never reaches them. At time period $j - 1$, the decryptor Bob holds $SK_{j-1} =$

$f(j-1)$. Via the aid of TA's, Bob would like to compute $SK_j = f(j)$, which shall be known to Bob only. Assume that each pair of Bob and TA's share a private channel by which secret information can be passed between them.

We treat Bob as TA_0 and let $x_0 = j-1$. By the Lagrange interpolation method, the polynomial that passes the shares of Bob and TA's is

$$f(x) = \sum_{k=0}^z (f(x_k) \cdot \prod_{0 \leq i \neq k \leq z} \frac{x - x_i}{x_k - x_i}).$$

TA_k can compute $s_k = f(x_k) \cdot \prod_{0 \leq i \neq k \leq z} \frac{j - x_i}{x_k - x_i}$, $0 \leq k \leq z$. Therefore, $f(j) = \sum_{k=0}^z s_k$. Our goal is that TA's together compute $f(j)$ and only TA_0 knows $f(j)$. Moreover, each TA_i does not reveal any information about its share $f(x_i)$. Note that x_0, x_1, \dots, x_z are known to all TA's. We describe the distributed protocol D-UPD for computing $f(j)$ securely as follows.

1. First, each TA_l , $1 \leq l \leq z$, selects a degree- z polynomial $h_l(x) = \sum_{i=1}^z a_{l,i}x^i + s_l$ over Z_q and sends $h_l(x_i)$ to TA_i , $0 \leq i \leq z$, via the private channel between them. Let $F(x) = \sum_{i=1}^z h_i(x)$.
2. Then, each TA_l , $0 \leq l \leq z$, computes its share $F(x_l) = \sum_{i=1}^z h_i(x_l)$.
3. Afterwards, each TA_l , $1 \leq l \leq z$, sends $F(x_l)$ to TA_0 via the private channel between them.
4. TA_0 then computes the constant coefficient $\sum_{i=1}^z s_i$ of $F(x)$ from $F(x_0)$, $F(x_1), \dots, F(x_z)$ by the Lagrange interpolation method and his private key $SK_j = f(j) = s_0 + \sum_{i=1}^z s_i$ at time period j .

Correctness. Following the protocol, D-UPD correctly computes the secret key SK_j . In the protocol, each TA_i does not have any information about another TA_j 's share $f(x_j)$.

We can make the computation verifiable by letting each TA_l publish $g^{a_{l,0}}, g^{a_{l,1}}, \dots, g^{a_{l,z}}$ [30]. Each TA_i , $0 \leq i \leq z$, verifies whether he receives the right share $h_l(x_i)$ from TA_l , $1 \leq l \leq z$, by checking $g^{h_l(x_i)} = \prod_{k=0}^z g^{a_{l,k}x_i^k}$.

4.1.2 Security analysis

We show that D-UPD is semantically secure against passive adversaries even z shares are disclosed.

Theorem 10 *Assume that the private channel exists. For D-UPD, given public key $PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z} \rangle$ and z private keys $SK_{j_1}, SK_{j_2}, \dots, SK_{j_z}$, no probabilistic polynomial-time adversary can compute another share SK_j in which $j \neq j_l, 1 \leq l \leq z$.*

Proof. We construct a simulator to simulate the procedure. If one can compute useful information from the transcripts of the real run, he can compute useful information from the transcripts of the simulation. Since the simulation does not disclose any useful information, an adversary can not compute any useful information from the real run. The simulation is described as follows.

Input: the shares s_1, s_2, \dots, s_t , the $PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z} \rangle$ and let $\mathcal{B} = \{B_1, B_2, \dots, B_t\}$ denote the corrupted set.

1. Each $B_i \in \mathcal{B}$, we select a degree- z polynomial $h_i(x) = \sum_{j=1}^z a_{i,j}x^j + s_i$ over Z_q . Then, B_i sends $h_l(x_i)$ to $B_j, 0 \leq j \leq z$.
2. Each $B_j \notin \mathcal{B}$, we randomly selects $h_j(x_0), \dots, h_j(x_{j-1}), h_j(x_{j+1}), \dots, h_j(x_{z+1})$ over Z_q . Then, B_j send $B_i h_j(x_i)$.
3. Then each $B_i \in \mathcal{B}$ computes the share $F(x_i) = \sum_{l=1}^z h_l(x_i)$ and send it to B_0 .

For $B_i \in \mathcal{B}$, since the transcript $(h_1(x_i), \dots, h_z(x_i))$ in the simulation consists of $Z + 1$ random values that are selected from Z_q , the transcript in the simulation is identical to that in the real run. Since the simulation carries no more useful information, the real procedure also does not disclose useful information. This completes the proof. \square

4.1.3 Proactivizing TA's shares

To protect TA's shares, We use proactive cryptography further. Let PS_i , $1 \leq i \leq n$, be n proactive servers. In practicality, we can make TA's as proactive servers. We proactivize each TA_i 's share $f(x_i)$ into PS_i 's. by the proactive (t, n) -secret sharing scheme [47]. Each evolving time period $j - 1$ is divided into sub-periods p_1, p_2, \dots, p_b , where p_1, p_2, \dots, p_{b-1} are refresh sub-periods in which the shares of $f(x_1), f(x_2), \dots, f(x_z)$ are refreshed among PS_i 's. In the last sub-period p_b , $f(j - 1)$ of the decryptor is updated to $f(j)$.

For simplicity, we let $t = n$. Let $r_i(x)$ be a degree- $(n - 1)$ polynomial over Z_q with constant coefficient $f(x_i)$, $1 \leq i \leq z$. Each PS_j , $1 \leq j \leq n$, holds a share $r_i(j)$ for the share $f(x_i)$, $1 \leq i \leq z$, and refreshes them in every sub-time period p_i , $1 \leq i \leq b - 1$, as follows.

1. Each PS_l , $1 \leq l \leq n$, selects n degree- $(n-1)$ polynomials $r_{l,i}(x)$ over Z_q , $1 \leq i \leq n$, whose constant coefficients are all 0. PS_l sends $r_{l,i}(j)$, $1 \leq i \leq n$, to PS_j , $1 \leq j \leq n$, via the private channel between them.
2. After receiving shares from other proactive servers, PS_l , $1 \leq l \leq n$, updates its shares to $r'_i(l) = r_i(l) + \sum_{j=1}^n r_{j,i}(l)$.

To update the decryptor Bob's decryption key $f(j - 1)$, we assume Bob as PS_0 and $x_0 = j - 1$. Let

$$\rho_{j,k} = \prod_{0 \leq i \neq k \leq z} \frac{j - x_i}{x_k - x_i} \text{ and } \lambda_l = \prod_{1 \leq i \neq l \leq n} \frac{-i}{l - i},$$

where $0 \leq k \leq z$ and $1 \leq l \leq n$. We have

$$\begin{aligned} f(j) &= \sum_{k=0}^z \rho_{j,k} f(x_k) = \sum_{k=1}^z \sum_{l=1}^n \rho_{j,k} \lambda_l r_k(l) + \rho_{j,0} f(x_0) \\ &= \sum_{l=1}^n \left(\sum_{k=1}^z \rho_{j,k} \lambda_l r_k(l) \right) + \rho_{j,0} f(x_0). \end{aligned}$$

Let $s_l = \sum_{k=1}^z \rho_{j,k} \lambda_l r_k(l)$, which can be computed by PS_l , $1 \leq l \leq n$. Our proactive key update scheme P-UPD for computing $SK_j = f(j)$ in the sub-period p_b of time period j is as follows.

1. Each PS_l , $1 \leq l \leq n$, selects a degree- n polynomial $h_l(x) = \sum_{i=1}^n a_{l,i}x^i + s_l$ over Z_q and sends $h_l(x_i)$ to PS_i , $0 \leq i \leq n$, via the private channel between them. Let $F(x) = \sum_{i=1}^n h_i(x)$.
2. Each PS_l , $0 \leq l \leq n$, computes its share $F(x_l) = \sum_{i=1}^n h_i(x_l)$.
3. Each PS_l , $1 \leq l \leq n$, sends $F(x_l)$ to PS_0 via the private channel between them.
4. PS_0 then computes the constant coefficient $\sum_{l=1}^n s_l$ of $F(x)$ from $F(x_0)$, $F(x_1), \dots, F(x_n)$ by the Lagrange interpolation method and his private key $SK_j = f(j) = \rho_{j,0}f(x_0) + \sum_{l=1}^n s_l$.

Correctness follows the above equations. Again, PS_i does not have any information about another PS_j 's shares $r_1(j), r_2(j), \dots, r_z(j)$. We can also make the computation verifiable by the verifiable secret sharing method as that in Section 4.1.1.

4.1.4 Security analysis

We show that P-UPD is semantically secure against passive adversaries even z shares are disclosed.

Theorem 11 *Assume that the private channel exists. For P-UPD, given public key $PK = \langle g^{a_0}, g^{a_1}, \dots, g^{a_z} \rangle$ and z private keys $SK_{j_1}, SK_{j_2}, \dots, SK_{j_z}$, no probabilistic polynomial-time adversary can compute the share SK_j in which $j \neq j_l, 1 \leq l \leq z$.*

Proof. We construct a simulator to simulate the procedure. If one can compute useful information from the transcripts of the real run, he can compute useful information from the transcripts of the simulation. Since the simulation does not disclose any useful information, an adversary can not compute any useful information from the real run. The simulation is described as follows.

Input: the shares s_1, s_2, \dots, s_n , and let $\mathcal{B} = \{PS_1, PS_2, \dots, PS_n\}$ denote the corrupted set.

1. Each $PS_i \in B$, we select a degree- n polynomial $h_i(x) = \sum_{j=1}^z a_{i,j}x^j + s_i$ over Z_q .
2. For $PS_j \notin B$, we randomly select z numbers $h_j(x_0), \dots, h_j(x_{j-1}), h_j(x_{j+1}), \dots, h_j(x_{z+1})$ over Z_q . Then, PS_j send PS_i $h_j(x_i)$.
3. Each $PS_i \in B$ computes its share $F(x_i) = \sum_{l=1}^n h_l(x_i)$.

For $PS_i \in B$, since the transcript $(h_1(x_i), \dots, h_n(x_i))$ is randomly selected over Z_q in the simulation, the transcript of the simulation and that in the real run are identical. Because the simulation carries no useful information, the real procedure also does not disclose useful information. This completes the proof. \square

4.2 Distributed KE-ENC schemes

It is sometimes desirable to have distributed decryption in which the decryption key is shared among a set of decryptors B_i , $1 \leq i \leq n$. Each decryptor holds a share s_i of the decryption key k . For decryption, each decryptor computes a partial plaintext $m_i = D(s_i, c)$ such that any one can combine these partial plaintexts to form the full plaintext m , where $c = E(k, m)$. We can easily make the decryption algorithms of our schemes work in a "distributive" way.

We assume that each decryptor B_i holds a share s_i of the secret key $SK_j = f(j)$ via a polynomial $t(x) = \sum_{k=1}^z a_k x^k + f(j) \pmod q$ such that $s_i = t(i)$. For KEENCBASIC, on receiving a ciphertext (j, α, s) , B_i computes the partial plaintext $m_i = \alpha^{s_i} \pmod p$. With $z + 1$ partial plaintexts $m_{i_1}, m_{i_2}, \dots, m_{i_{z+1}}$, one can compute the plaintext

$$m = s / \prod_{k=1}^{z+1} (m_{i_k})^{\lambda_k} \pmod p = s / \alpha^{f(j)} \pmod p,$$

where λ_k 's are appropriate Lagrange coefficients.

Correctness. Following the procedure, the correctness is easily verified.

Security analysis. Assume that the DDH problem is hard. If the single-user scheme KEENCBASIC is secure, the distributed KE-ENC scheme is semantically secure against passive adversaries even z shares are disclosed.

Theorem 12 *Assume that the DDH problem is hard. The distributed KE-ENC scheme is semantically secure against passive adversaries if the single-user scheme KEENCBASIC is secure.*

Proof. Since the encryption procedure in the distributed KE-ENC scheme is the same as that in the single-user scheme KEENCBASIC. Thus, the encryption procedure in the distributed KE-ENC scheme is semantically secure against the passive adversaries.

Furthermore, we need to prove that the transcript $\langle m_{i_1}, m_{i_2}, \dots, m_{i_{z+1}} \rangle$ disclose no useful information for the passive adversaries. Assume that DDH problem is hard. Given $\langle g, g^{s_1}, \dots, g^{s_{z+1}} \rangle$, then the following two transcripts are polynomial time indistinguishable.

1. $D = \langle (g, g^k), (g^{s_1}, g^{ks_1}), \dots, (g^{s_{z+1}}, g^{ks_{z+1}}) \rangle$.
2. $R = \langle (g, g^k), (g^{s_1}, g^{c_1}), \dots, (g^{s_{z+1}}, g^{c_{z+1}}) \rangle$.

Since the passive adversaries cannot get more information from the transcript R , then the passive adversaries cannot get no more information from the decryption transcript D . This completes the proof. \square

4.3 Distributed KE-ENC with proactive security

The proactive key update scheme P-UPD of Section 4.1.3 can be easily adapted to the distributed case. Let DP-UPD denote the scheme. We treat proactive servers as decryptors B_1, B_2, \dots, B_n . The only amendment is to distributed the decryption key $f(j)$ of TA_0 among all decryptors. Let B_i , $1 \leq i \leq n$, be the

set of decryptors. At period $j - 1$, each of $f(j - 1), f(x_1), f(x_2), \dots, f(x_z)$ is shared among B_i 's. Each evolving time period j is divided into sub-periods p_1, p_2, \dots, p_b , where p_1, p_2, \dots, p_{b-1} are refresh sub-periods in which the shares of $f(j - 1), f(x_1), f(x_2), \dots, f(x_z)$ are refreshed. In the last sub-period p_b , shares of $f(j - 1)$ are updated to shares of $f(j)$ and then time enters the period j .

Theorem 13 *Assume that the private channel exists. For DP-UPD, an adversary \mathcal{A} can corrupt z TAs at most and obtain all secrets held by z TAs. The new share $f(j)$ is still unknown to the passive adversary \mathcal{A} .*

Proof. We construct a simulation as follows. Let $\mathcal{B} = \{B_1, B_2, \dots, B_t\}$ denote the corrupted set. Let the secret $f(i)$ held by B_i . Each B_i holds the shares $\{s_{0,i}, \dots, s_{z,i}\}$. Each $B_i \in \mathcal{B}, 1 \leq i \leq t$, selects z degree- z polynomials $h_{i,c}(x) = \sum_{k=1}^z a_{i,k}x^k + s_{i,c}$ over Z_q for $c = 1, \dots, z$ and sends $\{h_{i,1}(t), \dots, h_{i,z}(t)\}$ to $B_t, 0 \leq t \leq z$, via the private channel between them. For $B_j \notin \mathcal{B}$, we randomly selects $h_{j,c}(x_0), \dots, h_{j,c}(x_{j-1}), h_{j,c}(x_{j+1}), \dots, h_{j,c}(x_{z+1})$ over Z_q for $c = 1, \dots, z$. Then, B_j sends $B_i \{h_{j,1}(x_i), \dots, h_{j,z}(x_i)\}$ via the private channel between them. Let $H_{i,c}(x) = \sum_{i=0}^z h_{i,c}(x)$. The new share held by B_j is $s_{i,j} = H_{i,j}(x_j)$ for $i = 1, \dots, z$. For $B_j \in \mathcal{B}$, since the transcripts $\{h_{j,1}(x_j), \dots, h_{j,z}(x_j)\}$ for $j = 1, \dots, z$ are randomly selected, the shares $s_{i,j}$ for $j = 1, \dots, z$ are random values. Thus, the distribution of real run is indistinguishable from the distribution of the simulation. If one can compute any useful information from the outputs of the simulation, he can obtain any useful information from the real run. However, since the transcripts of simulation cannot have any useful information, that of the real run also cannot have any useful information. \square

Chapter 5

Threshold Forward-Secure Signature Schemes

In this chapter, we propose one threshold forward-secure signature scheme. Our scheme is the modification of the forward-secure signature scheme of Abadalla and Reyzin [4]. We discuss how to achieve to threshold forward-secure signature scheme and the security of our scheme.

Proactive cryptography combines the concepts of "distributing the secret" and "refreshing the shares" to provide security against the mobile adversary, who attacks the parties of a distributed cryptosystem dynamically. For an adversary, we cannot assume that it cannot break into a particular party, who holds a share of the secret, during the party's lifetime. However, we can assume that the adversary can break into at most t parties during a short period of time, say an hour. Based on this observation, the proactive cryptography "refreshes" each party's share periodically. It divides the time into several time periods, starting at 0. At the end of each time period, there is a "refresh phase" during which each party refreshes its share, but the secret they share remains intact. We assume that the mobile adversary can corrupt all parties during the lifetime of the cryptosystem; nevertheless, it can corrupt at most t parties during a time period. The proactive mechanism provides a high level of security for cryptosystems so that we would like to proactivize important

cryptographic primitives.

In the chapter we propose a protocol that proactivizes the forward-secure signature scheme of Abadalla and Reyzin [4]. The Abadalla and Reyzin's forward-secure signature scheme is an improvement of the Bellare-Miner scheme [13] with a shorter public key. Abadalla, et. al. has proactivized the Bellare-Miner forward-secure signature scheme [3]. They proposed two threshold signature schemes in proactivizing Bellare-Miner forward-secure signature scheme. One scheme uses multiplicative secret sharing and the other uses polynomial secret sharing. In our scheme, we combine both secret sharing schemes for efficiency. The trick of multiplicative secret sharing is used to sign a message in a distributed way and polynomial secret sharing is used to share the signing secret. Our scheme is not only robust, but also efficient.

It is worth mentioning that we propose a new scheme for multiplying two secrets that are shared among parties [11, 17, 44]. Our multiplication scheme is efficient since it uses the public channel and the private channel once only.

Communication model. We assume that the involved n parties are connected by a broadcast channel such that the messages over the channel cannot be blocked, delayed or altered. Nevertheless, one can inject false messages. Any two parties are connected by a private channel such that a third party cannot get messages sent over the private channel. We also assume that the communication channel is synchronous by rounds, that is, all parties send messages simultaneously in a round.

Time. There is a universal clock such that each party knows the absolute time. Therefore, we can divide time into time periods, starting at 0. Each time period has two phases: the execution phase and the refresh phase. The refresh phase follows the execution phase. The parties sign messages during the execution phase. During the refresh phase, all parties together run the share refresh algorithm to refresh their shares.

Adversary. We consider the static adversary who chooses corrupted parties at the beginning of each time period. The adversary runs three phases: the

chosen message attack phase (CMA), the break-in phase (BREAKIN), and the forgery phase (FORGE). The BREAKIN phase for the threshold signature scheme is equivalent to the OVERTHRESHOLD phase of [3].

In the CMA phase, the adversary can corrupt at most t parties for any period of time. The adversary gets all information in the corrupted parties, including their shares, random bits, etc. The adversary can query the signing oracle \mathcal{S}_x , where x is the secret signing key. Since we assume the random oracle model [12], the adversary is allowed to query the random oracle \mathcal{H} corresponding to the collision-resistant hash function used in the scheme. At the end of the CMA phase, the adversary can stay in the current phase or enter the next BREAKIN phase. In the BREAKIN phase, the adversary can corrupt *more than* t parties. Let c be the period that the adversary enters the BREAKIN phase and corrupts more than t users. In this phase, the adversary can compute the master secret (the signing key) of period c from the shares of corrupted parties. Then, the adversary enters the FORGE phase, during which the adversary outputs a forged signature of a new message which has not been queried to the signing oracle. We say that the adversary succeeds in attacking the scheme if it outputs a forged signature for a prior time period c' , $c' < c$, with non-negligible probability.

5.1 Building blocks

The following system setting is used throughout the rest of the chapter.

- Let $p=4p'q' + 1$ be a prime, where p' and q' are large primes and $p' \equiv q' \equiv 3 \pmod{4}$.
- Let $N = p'q'$ and g a generator of the order- N subgroup of Z_p^* . All operations hereafter will be over the order- N subgroup, unless stated otherwise.
- The involved parties are dealers D_i , $1 \leq i \leq n$.

(t, n) -VSS PROCEDURE. If dealer D_i wants to share a random secret with other dealers, it runs the following steps.

1. Select a random polynomial $f_i(x)$ of degree t over Z_N^* . The constant coefficient of $f_i(x)$ is the random secret.
2. Send share $f_i(j)$ to dealer D_j , $j \neq i$ via the private channel.
3. Publish the verification values $\langle g^{a_{i,0}}, \dots, g^{a_{i,t}} \rangle$.
4. Dealer D_j verifies validity of its received share $f_i(j)$ by $\prod_{k=0}^{k=t} g^{a_{i,k} j^k} = g^{f_i(j)}$.

If the verification fails, D_j requests D_i to publish $f_i(j)$. If D_i does not cooperate or posts an inconsistent $f_i(j)$, D_i is disqualified.

RECOVERY PROCEDURE. We use Lagrange's interpolation method to recover the secret with at least $t + 1$ shares.

PROOF-SS PROCEDURE. Given (g, t, N, F, T) , prover P wants to convince verifier V two things: (1) $a = \log_g F \bmod p = T^{1/t} \bmod N$ and (2) it knows this a . This is a combination of proofs of membership and knowledge.

1. The prover P selects random $w \in Z_N^*$ and sends $H = F^w$ and $B = w^t \bmod N$ to V .
2. The verifier V selects a random challenge $c \in \{0, 1\}$ and sends it to P .
3. The prover P sends $r = a^c w \bmod N$ to V .
4. The verifier V checks (1) $H = F^r$ and $B = r^t \bmod N$ if $c = 0$; and (2) $H = g^r$ and $B = r^t / T \bmod N$ if $c = 1$.

We use $\text{PROOF-SS}(g, t, g^a, a^t)$ to denote the above interactive proof system.

Theorem 14 *PROOF-SS is complete, sound and zero-knowledge.*

Proof. The completeness property can be verified easily. For soundness of proof of knowledge, if any prover P^* can convince V with a non-negligible

probability ϵ , P^* and V together can compute a with an overwhelming probability. Let (W, C, R) denote the random variables for the real view. Then, $Pr(W, C, R) \geq \epsilon$. By a probabilistic argument, there is a subset Λ of W of probability $\epsilon/2$, that is, $Pr_{w \in \Lambda}(w, C, R) \geq \epsilon/2$. Also, P^* can answer two different challenges c_1 and c_2 with probability $\epsilon/2$. That is, $Pr_{w \in \Lambda}(w, c_1, r_1) \geq \epsilon/2$ and $Pr_{w \in \Lambda}(w, c_2, r_2) \geq \epsilon/2$ where $c_1, c_2 \in C$ and $r_1, r_2 \in R$. Therefore, we can get two responses $r_1 = a^{c_1}w \bmod N$ and $r_2 = a^{c_2}w \bmod N$ for the same commitments H and B . We can compute $a = r_2/r_1 \bmod N$ assuming, without loss of generality, $c_1 = 0$ and $c_2 = 1$. For soundness of proof of membership, we can easily show that if F and T are not of right form, the probability that P^* can cheat V is 0.5 (and is negligible after a polynomial number of rounds.)

For zero-knowledge, we construct a simulator S to simulate the view of any verifier V^* . M first selects $c' \in \{0, 1\}$ and $r \in Z_N^*$ randomly and computes $H = F^r$ and $B = r^t \bmod N$ if $c' = 0$ and $H = g^r$ and $B = r^t/T \bmod N$ if $c' = 1$. S then simulates $V^*(H, B)$ to get c . If $c = c'$, M outputs (H, B, c, r) ; otherwise M outputs \perp . The output of M conditioned on that the output is not \perp and the view of V^* are statistically indistinguishable. Since V^* does not know M 's selection c' , $Pr[C'' = c] = 1/2$ where C'' denote the random variable in simulation. Let $\{\langle P, V^* \rangle(H, B, C, R)\}$ denote the real view and $\{M_{V^*}(H, B, C, R)\}$ denote the view of the simulator M . With the constraints $h_0 = F^{r(1-c)} \cdot g^{rc}$ and $b_0 = r^t/T^c \bmod N$, for any fixed h_0, b_0, c_0, r_0 , we have

$$Pr[(H, B, C, R) = (h_0, b_0, c_0, r_0)] = \frac{Pr[\tilde{V}(h_0, b_0) = c_0]}{p' \cdot q'}$$

Let (H', B', C', R') denote the random variables for the simulated view produced by M^* . For the above (h_0, b_0, c_0, r_0) , we have

$$\begin{aligned} & Pr[m^*(V^*, (g, t, N, F, T)) = ((g, t, N, F, T), h_0, b_0, c_0, r_0)] \\ &= Pr[(H', B', C', R') = (h_0, b_0, c_0, r_0) | M^*(V^*, (g, t, N, F, T)) \neq \perp] \\ &= \frac{Pr[\tilde{V}(h_0, b_0) = c_0, C'' = c_0, M^*(V^*, (g, t, N, F, T)) \neq \perp]}{p' \cdot q' \cdot Pr[M^*(V^*, (g, t, N, F, T)) \neq \perp]} \end{aligned}$$

$$\begin{aligned}
&= \frac{\Pr[\tilde{V}(h_0, b_0) = c_0, C'' = c_0]}{p' \cdot q' \cdot (1/2)} \\
&= \frac{\Pr[\tilde{V}(h_0, b_0) = c_0] \cdot \Pr[C'' = c_0]}{p' \cdot q' \cdot (1/2)} \\
&= \frac{\Pr[\tilde{V}(h_0, b_0) = c_0]}{p' \cdot q'}
\end{aligned}$$

Therefore, PROOF-SS is zero-knowledge. \square

We convert PROOF-SS into a non-interactive version by using a collision-resistant hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^l$ to replace V [31, 86], where l be the security parameter. The message (c, r_1, \dots, r_l) sent by P for non-interactive PROOF-SS, denoted by NIPROOF-SS, satisfies

$$c = \mathcal{H}(t||g||N||F||T||H_1||B_1||\dots||H_l||B_l),$$

where $||$ is the concatenation operator of strings. Let c_i denote the i -th bit of c . If c_i is 1, $H_i = g^{r_i}$ and $B_i = r_i^t/T \bmod N$; otherwise $H_i = F^{r_i}$ and $B_i = r_i^t \bmod N$. P can compute (c, r_1, \dots, r_l) by choosing $w_i \in Z_N^*$ for $i = 1, \dots, l$, computing $c = \mathcal{H}(t||g||N||F||T||F^{w_1}||w_1^t||\dots||F^{w_l}||w_l^t)$, and setting $r_i = a^{c_i}w_i \bmod N$. NIPROOF-SS releases no useful information under the random oracle model assuming hardness of discrete logarithm and factoring.

PROOF-DH PROCEDURE. Given (g, H, F) and the prover P wants to convince V that $H = g^s$ and $F = g^{s^2}$ are of right form and it knows the secret s . The interactive proof system is as follows [22].

1. P randomly selects $w \in Z_N^*$ and sends $A = g^w$ and $B = H^w$ to V .
2. V sends a random challenge $c \in \{0, \dots, 2^k - 1\}$ to P .
3. P sends the response $r = w + cs \bmod N$ to V .
4. V checks $g^r = A \cdot H^c$ and $H^r = B \cdot F^c$.

The above PROOF-DH procedure is complete, sound, and zero knowledge. We use NIPROOF-DH to denote its non-interactive version.

SQ PROCEDURE. Let $h(x)$ be a degree- t polynomial over Z_N^* with $h(0) = s$ and shared by the dealers D_i , $1 \leq i \leq n$. SQ's goal is to make the dealers share a degree- t polynomial $h'(x)$ over Z_N^* with $h'(0) = s^2 \bmod N$. SQ procedure is as follows.

1. Dealer D_i selects two degree- t polynomials $f_i(x)$ and $e_i(x)$ over Z_N^* at random, where $e_i(0)=0$. It sends shares $f_i(j)$ and $e_i(j)$ to D_j via the private channel, $1 \leq j \leq n$. Using (t, n) -VSS PROCEDURE, D_j checks if the received shares are correct. If so, all dealers share two degree- t polynomial $F(x) = \sum_{i=1}^n f_i(x) \bmod N$ and $E(x) = \sum_{i=1}^n e_i(x) \bmod N$. Each dealer D_i holds shares $F(i)$ and $E(i)$.
2. Each dealer D_i publishes $u_i = h(i)^2 + F(i) \bmod N$ and NIPROOF-DH($g, g^{h(i)}, g^{h(i)^2}$) and checks validity of the published values of other dealers by checking $g^{u_j} = g^{h(j)^2} \cdot \prod_{k=0}^t g^{A_k j^k}$, where $g^{A_0}, g^{A_1}, \dots, g^{A_t}$, computed from the verification values of $f_i(x)$'s, are the verification values of $F(x)$.
3. Each dealer D_i computes the degree- $2t$ polynomial $T(x)=h(x)^2 + F(x) = \sum_{k=0}^{2t} t_k x^k$ over Z_N from u_j , $1 \leq j \leq n$. Let $T''(x) = \sum_{k=0}^t t_k x^k$, which is $h''(x) + F(x) \bmod N$ for some degree- t polynomial $h''(x)$. Note that $h''(0) = h(0)^2 \bmod N$.
4. Each dealer D_i computes its share $h''(i) = T''(i) - F(i) \bmod N$ and randomizes it to become $h'(i) = h''(i) + E(i)$. The hidden polynomial becomes $h'(x) = h''(x) + E(x) \bmod N$ whose constant coefficient is still $s^2 \bmod N$.

We use $\text{SQ}(C, h(x), h'(x))$ to denote the above procedure, where C is the dealer set, $h(x)$ is the shared polynomial initially and $h'(x)$ is the shared polynomial at the end.

Theorem 15 *SQ PROCEDURE is correct, robust and secure if there are at most $n/3$ corrupted dealers.*

Proof. We can check correctness easily. Since there are at most t corrupted dealers, $t < n/3$, honest dealers can smoothly finish the procedure. This is guaranteed by the (t, n) -VSS procedure.

We present a simulator to show that a malicious adversary, who corrupts at most t dealers, gets no information. Let \mathcal{B} be the corrupted set of dealers.

Input: $\langle g^s, g^{a_1}, \dots, g^{a_t} \rangle, h(i)$ for every dealer $D_i \in \mathcal{B}$, where $h(x) = s + \sum_{k=1}^t a_k x^k$;

1. Randomly select degree- t polynomials $\hat{f}_i(x)$ and $\hat{e}_i(x)$ with $\hat{e}_i(0) = 0$, $1 \leq i \leq n$. Let $\hat{F}(x) = \sum_{i=1}^n \hat{f}_i(x)$ and $\hat{E}(x) = \sum_{i=1}^n \hat{e}_i(x)$.
2. Run (t, n) -VSS PROCEDURE.
3. For each $D_i \notin \mathcal{B}$, randomly select \hat{u}_i over Z_N^* , compute $g^{h(i)^2} = g^{\hat{u}_i} / g^{\hat{F}(i)}$, and simulate NIPROOF-DH($g, g^{h(i)}, g^{h(i)^2}$), where $g^{h(i)} = g^s \cdot \prod_{j=1}^t g^{a_j i^j}$.
4. For each $D_i \in \mathcal{B}$, publish $u_i = h(i)^2 + \hat{F}(i) \bmod N$ and simulate NIPROOF-DH($g, g^{h(i)}, g^{h(i)^2}$).

In the real run, the transcripts among dealers include $2n$ degree- t polynomials, u_i s and *NIProof* – *DH*s. In the simulator, the transcripts also include $2n$ degree- t polynomials randomly selected over Z_N^* that is identical to that in the real. Since in the simulator u_i s is randomized by a random number $\hat{F}(i)$, the distribution of u_i s is also identical to that in the real. For *NIProof* – *DH*_s in the simulator except the status of failures the distribution of *NIProof* – *DH*_s is the identical to that in the real. The above simulation produces a distribution computationally indistinguishable from that of the real run. \square

Assume that the dealers share two degree- t polynomial $h_1(x)$ and $h_2(x)$ initially. We can modify the SQ procedure so that the dealers share a degree- t polynomial $h'(x)$ whose constant coefficient is $h_1(0)h_2(0) \bmod N$ at the end. Let MULT($C, h_1(x), h_2(x), h'(x)$) denote the procedure of sharing a degree- t polynomial $h'(x)$ whose constant coefficient is $h_1(0)h_2(0) \bmod N$.

5.2 Threshold forward-secure signature scheme

Our threshold forward-secure signature scheme, denoted by **TFSS**, is a key-evolving (t, s, n) -threshold signature scheme that consists of four procedures: **TFSS.KEY**, **TFSS.UPDATE**, **TFSS.SIGN**, and **TFSS.VERIFY**, where t is the maximum number of corrupted dealers, s is the minimum number of alive dealers so that signature computation is possible, and n is the total number of dealers. In our scheme, we set $s = t + 1$ and $n \geq 2t + 1$. There is a manager presiding the scheme.

TFSS.KEY. It takes as input a security parameter l and outputs the public key and initial secret-key share $S_{i,0}$ and public-key share $PK_{i,0}$ of the dealer D_i 's.

1. Select N as that in the system setting.
2. The manager randomly selects $S_{i,0} \in Z_N^*$, $1 \leq i \leq n$ and computes $U_{i,0} = 1/S_{i,0}^{2^{t(T+1)}} \bmod N$, $S_0 = \prod_{i=1}^n S_{i,0} \bmod N$, and $U = 1/S_0^{2^{t(T+1)}} \bmod N$.
3. The system's initial secret key is $SK_0 = (N, T, 0, S_0)$ and the public key $PK = (N, U, T)$.
4. Each dealer D_i 's initial secret-key share is $SK_{i,0} = (N, T, 0, S_{i,0})$ and public-key share is $PK_{i,0} = (N, U_{i,0}, T)$.
5. Each dealer D_i shares its $S_{i,0}$ with other dealers by the (t, n) -VSS PROCEDURE.

TFSS.UPDATE. At the end of time period j , all dealers take part in the procedure to update their shares. Each dealer updates its secret-key and public-key shares from $S_{i,j}$ and $PK_{i,j}$ to $S_{i,j+1}$ and $PK_{i,j+1}$.

1. Each dealer D_i randomly selects $n - 1$ numbers $s_{i,1}, s_{i,2}, \dots, s_{i,n-1}$ over Z_N^* and computes $s_{i,n} = S_{i,j} / \prod_{k=1}^{n-1} s_{i,k} \bmod N$.

2. Each dealer D_i sends $s_{i,k}$ to D_k privately and publishes $\hat{s}_{i,k} = 1/s_{i,k}^{2^{l(T+1-j)}} \bmod N$, $1 \leq k \leq n$.
3. Each dealer D_k checks validity of the published values by $U_{i,j} = \prod_{r=1}^n \hat{s}_{i,r} \bmod N$, $1 \leq i \leq n, i \neq k$. Dealer D_k also checks validity of its received secret $s_{i,k}$ by $1/s_{i,k}^{2^{l(T+1-j)}} \bmod N = \hat{s}_{i,k}$. If one of the checks fails, all other dealers recover the secret $S_{i,j}$ by RECOVERY PROCEDURE.
4. Dealer D_i 's new secret-key share is $S_{i,j+1} = (\prod_{k=1}^n s_{k,i})^{2^l} \bmod N$ and the corresponding public-key share is $U_{i,j+1} = \prod_{k=1}^n \hat{s}_{k,i} \bmod N$.
5. Dealer D_i shares $S_{i,j+1}$ with other dealers by (t, n) -VSS PROCEDURE. We use NIPROOF-SS($g, t, g^{S_{i,j+1}}, S_{i,j+1}^t$) to verify whether D_i 's action is correct, where $t = -2^{l(T-j)}$ and $S_{i,j+1}^t = U_{i,j+1}$. If the proof is correct and (t, n) -VSS PROCEDURE succeeds, all dealers delete their old secret-key shares; otherwise, the secret of D_i is reconstructed.

TFSS.SIGN: at time period j , all dealers sign a messages M in a distributed way with the following steps.

1. Each dealer D_i selects $R_i \in Z_N^*$ randomly and publishes $Y_i = R_i^{2^{l(T+1-j)}} \bmod N$ and NIPROOF-SS($g, 2^{l(T+1-j)}, g^{R_i}, Y_i$). Then, it shares R_i to other dealers via (t, n) -VSS PROCEDURE with polynomial $f_i(x)$. If NIPROOF-SS or (t, n) -VSS PROCEDURE fails, set $R_i = 1$ and run RECOVERY PROCEDURE to recover the secret-key share $S_{i,j}$ of D_i .
2. Each dealer D_i computes $Y = \prod_{i=1}^n Y_i$ and $\sigma = \mathcal{H}(j, Y, M)$ and publishes its partial signature $Z_i = R_i S_{i,j}^\sigma \bmod N$.
3. Each dealer D_i verifies validity of another dealer D_k 's partial signature by computing

$$Y_i' = Z_i^{2^{l(T+1-j)}} U_{i,j}^\sigma \bmod N$$

and checking whether Y_i' and Y_i are equal. If the verification fails, all other alive dealers run RECOVERY PROCEDURE to recover the secret-key share $S_{k,j}$ and R_k of D_k and compute the partial signature Z_k .

4. Combine all partial signatures as a signature (j, Z, σ) for M at time j , where $Z = \prod_{i=1}^n Z_i \bmod N$. All dealers erase their R_i 's.

TFSS.VERIFY: We can use the public key $PK = (N, U, T)$ of the system to verify validity of a signature (j, Z, σ) for M .

1. If $Z = 0$, return '0'.
2. Otherwise, compute $Y' = Z^{2^{l(T+1-j)}} U^\sigma \bmod N$ and output '1' if and only if $\sigma = H(j, Y', M)$.

5.3 Security analysis

In this section, we show the correctness and security of our proposed scheme.

Theorem 16 (Correctness) *Assume that $SK_j = (N, T, j, S_j)$ and $PK = (N, U, T)$ are key pairs of the system at time period j . Each dealer D_i holds the secret-key share $SK_{i,j} = (N, T, j, S_{i,j})$ and public-key share $PK_{i,j} = (N, U_{i,j}, T)$. If (j, Z, σ) is generated by TFSS.SIGN for M , $\text{TFSS.VERIFY}(PK, j, Z, \sigma) = 1$.*

Proof. We have $S_j = \prod_{i=1}^n S_{i,j} \bmod N$, $U = \prod_{i=1}^n U_{i,j} \bmod N = \prod_{i=1}^n S_{i,j}^{-2^{l(T+1-j)}} \bmod N$, $Y = \prod_{i=1}^n R_i^{2^{l(T+1-j)}} \bmod N = \prod_{i=1}^n Y_i \bmod N$ and $Z = \prod_{i=1}^n Z_i \bmod N = \prod_{i=1}^n R_i S_{i,j}^\sigma \bmod N$. Since

$$\begin{aligned}
Y' &= Z^{2^{l(T+1-j)}} U^\sigma \bmod N = \prod_{i=1}^n (R_i S_{i,j}^\sigma)^{2^{l(T+1-j)}} \prod_{i=1}^n U_{i,j}^\sigma \bmod N \\
&= \prod_{i=1}^n [R_i^{2^{l(T+1-j)}} S_{i,j}^{\sigma 2^{l(T+1-j)}} U_{i,j}^\sigma] \bmod N = \prod_{i=1}^n R_i^{2^{l(T+1-j)}} \bmod N \\
&= \prod_{i=1}^n Y_i \bmod N = Y,
\end{aligned}$$

we have $\mathcal{H}(j, Y', M) = \mathcal{H}(j, Y, M) = \sigma$. □

Theorem 17 *Assume that 2^l -th square root problem is difficult. TFSS.UPDATE procedure is secure against malicious adversaries even malicious adversaries know t shares, s_{b_i} for $1 \leq i \leq t$.*

Proof. Given t shares, if there exists a malicious adversary \mathcal{A} can compute any useful information from the transcripts among the dealers. We can construct a simulator S to simulate the procedure such that we can use A and the outputs of the simulator to compute useful information. Since the distribution of the simulator and that in the real run are polynomial indistinguishable, if the adversary A can compute any information from that in the real run, he can compute any useful information from the outputs of the simulator. Because the output of simulator carries no information, the output of the real run also carries no information. We construct a simulator S to simulate TFSS.UPDATE procedure assuming existence of malicious adversaries. Let $\mathcal{B} = \{D_{b_1}, \dots, D_{b_t}\}$ be the set of corrupted servers at current time j . For simplicity, the secrets of corrupted dealers are treated as inputs. S simulates each dealer D_i 's behavior as follows.

Input: $PK = (N, U, T)$, secret keys $S_{b_k, j}, 1 \leq k \leq t$, shares $\hat{f}_k(b_i), 1 \leq i \leq t, 1 \leq k \leq n$, $PK_{i, j} = (N, U_{i, j}, T), 1 \leq i \leq n$, and $\langle g^{S_{i, j}}, g^{a_{i, 1}} \dots, g^{a_{i, t}} \rangle, 1 \leq i \leq n$;

1. Randomly select $\hat{s}_{i, 1}, \dots, \hat{s}_{i, (i-1)}, \hat{s}_{i, (i+1)}, \dots, \hat{s}_{i, n-1}$ from Z_N^* , compute

$$1/(\hat{s}_{i, i}^{2^{l(T+1-j)}}) = U_{i, j} / \prod_{k=1, k \neq i}^n 1/(\hat{s}_{i, k}^{2^{l(T+1-j)}}) \pmod N,$$

and publish $\hat{S}_{i, k} = 1/(\hat{s}_{i, k}^{2^{l(T+1-j)}}) \pmod N$ for $k = 1, \dots, n$. Note that we do not know the value $\hat{s}_{i, i}$ for $D_i \notin \mathcal{B}$.

2. Randomly select polynomial $\hat{h}_i(x)$ over Z_N^* . Let $\hat{h}_i(0) = S'_{i, j+1}$, which is a random value in Z_N^* for $D_i \notin \mathcal{B}$. Simulate NIPROOF-SS($g, -2^{l(T-j)}, g^{S'_{i, j+1}}, U_{i, j+1}$), where $U_{i, j+1} = \prod_{k=1}^n \hat{S}_{k, i} \pmod N$.
3. For $D_i \in \mathcal{B}$, compute its new secret-key share $S_{b_k, j+1}$ by $\prod_{i=1}^n \hat{s}_{i, b_k}^{2^l} \pmod N$ and simulate NIPROOF-SS($g, -2^{l(T+1-(j+1))}, g^{S_{b_k, j+1}}, U_{b_k, j+1}$), where $U_{b_k, j+1} = \prod_{i=1}^n \hat{S}_{i, b_k} \pmod N$.

4. Simulate (t, n) -VSS PROCEDURE. For $D_i \notin \mathcal{B}$, since $S'_{i,j}$ of $g^{S'_{i,j}}$ is unknown, we can randomly select $S'_{i,k} \in_R Z_N^*$, $b_1 \leq k \leq b_t$. Then, $(j, S'_{i,j})$, $(b_1, S'_{i,b_1}), \dots, (b_t, S'_{i,b_t})$ can construct a polynomial function with degree- t even $S'_{i,j}$ is unknown. We can do it by the method of Lagrange interpolation in exponentiation. We send each S'_{i,b_i} to the corrupted dealer D_{b_i} .

If D_j forces D_i to disclose $\hat{s}_{i,j}$, since D_j has it, we can simulate $\hat{s}_{i,j}$.

We consider that the distribution of the outputs for the simulator. In the step one, each $\hat{s}_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq n$, is randomly selected from Z_N^* such that the distribution of that is identical to that in the real run. In the step two, if the $D_i \in \mathcal{B}$, the distribution of the transcripts of D_i is the same as that in the real one. If the $D_i \notin \mathcal{B}$, except the failure, i.e., \perp , the distribution of the transcripts of D_i is identical to that in the real one. In the step three, (t, n) -VSS PROCEDURE outputs the transcripts of the n polynomial functions. The distribution of the transcripts is identical to that of the transcripts in the real run since each coefficient of polynomial functions is randomly selected over Z_N^* . Therefore, the distribution of the outputs for the simulator is polynomial indistinguishable to the distribution of that in the real run. Since the outputs of simulation gives no information, that in the real run also gives no information. The adversary cannot get useful information. This completes the proof. \square

Theorem 18 *The TFSS scheme is a key-evolving (t, s, n) -threshold signature scheme for $s = t + 1$ and $n = 2t + 1$.*

Proof. Since there are at most t corrupted servers, their secret-key shares are not sufficient to recover the secret-key shares of honest dealers. The others follow the scheme. \square

Theorem 19 (Forward secrecy) *Let FS-DS denote the single-user signature scheme in [4]. TFSS is a threshold forward-secure signature scheme as long as FS-DS is a forward-secure signature scheme in the single-user sense.*

Proof. Let \mathcal{F} be the adversary who attacks TFSS successfully by forging a signature (c', Z, α) . We construct an algorithm \mathcal{A} that uses this \mathcal{F} to forge a signature for the single-user FS-DS. As stated, the attacking procedure contains three phases: CMA, BREAKIN, and FORGE. The algorithm \mathcal{A} contains the signing oracle \mathcal{S} and the hashing oracle \mathcal{H} , which can be allowed to query and provide the signing key if necessary. In the CMA phase, \mathcal{F} can query signatures and hash values from the signing oracle \mathcal{S} and the hashing oracle \mathcal{H} . Given a signature, we can simulate the transcripts as that in the real view. In the BREAKIN phase, the signing oracle provides \mathcal{F} with the secret key SK_j such that \mathcal{F} can forge a signature σ of the time period t , $t < j$ in the FORGE phase. Algorithm \mathcal{A} outputs \mathcal{F} 's output σ as the forged signature of the single-user scheme. We simulate the procedure as follows.

In the CMA phase, \mathcal{F} guesses a particular time period c during which \mathcal{F} breaks more than t dealers and gets the secret S_c . Let $U = 1/v^{2^{l(T+1-c)}}$ and $PK = (N, U, T)$, where $v = S_c$. We randomly select $U_{i,0}, \dots, U_{n-1,0} \in_R Z_N^*$ and compute public-key share $U_{n,0} = U / \prod_{i=1}^{n-1} U_{i,0} \bmod N$. The public key is $PK_{i,0} = (N, U_{i,0}, T)$, $1 \leq i \leq n$. We simulate \mathcal{F} by choosing a random tape for \mathcal{F} , feeding all public keys to \mathcal{F} , and running F in the CMA phase. \mathcal{F} can corrupt at most t dealers at any time period except the time period c . Since \mathcal{F} can corrupt at most t dealers at any time period except at time period c , we simply give all necessary secret-key shares and exchanged shares as \mathcal{F} 's input. \mathcal{F} decides either to stay at the CMA phase or to switch to the BREAKIN phase, and then enter the FORGE phase.

We now we simulate the views of corrupted dealers during the key update phase. Let $\mathcal{B} = \{D_{b_1}, \dots, D_{b_t}\}$ be the set of corrupted dealers at time period j . The simulation is the same as that of Theorem 17, which simulates the key update procedure. Note that the set of corrupted servers is decided in advance.

We can simulate the hash and signing oracles of \mathcal{F} . For each query (j, Y, M) made by \mathcal{F} , we query \mathcal{H} on the same input and return the answer to \mathcal{F} . We simulate the signing oracle of \mathcal{F} by using \mathcal{S} . Let M be the message queried to

\mathcal{S} . We give the direct answer (j, Z, σ) of S to F .

Now, we simulate \mathcal{F} 's view of the signing procedure. The input consists of all secrets of the corrupted dealers and public information. For the input M and its signature (j, Z, σ) seen by \mathcal{F} , we construct the same probability distribution of \mathcal{F} 's real view as follows.

1. For $D_i \in \mathcal{B}$, we directly choose $R_i \in Z_N^*$ and publish $Y_i = R_i^{2^{l(T+1-j)}} \bmod N$ and NIPROOF-SS($g, 2^{l(T+1-j)}, g^{R_i}, Y_i$). Then, we simulate (t, n) -VSS PROCEDURE to share R_i with other dealers. Furthermore, we compute the partial signature $Z_i = R_i S_{i,j}^\sigma \bmod N$.
2. For $D_i \notin \mathcal{B}$, we compute its partial signature as follows. Let $Z' = Z / \prod_{i=1}^t Z_i \bmod N$. We randomly select $n-t-1$ numbers from Z_N^* , says $Z_{c_1}, \dots, Z_{c_{n-t-1}}$. We compute $Z_{c_{n-t}} = Z' / \prod_{i=1}^{n-t-1} Z_{c_i} \bmod N$.
3. We compute $Y_{c_i} = Z_{c_i}^{2^{l(T+1-j)}} U_{c_i,j}^\sigma \bmod N$ for $1 \leq i \leq n-t$, and randomly select $(n-t)$ numbers from Z_N^* , says $R_{c_1}, \dots, R_{c_{n-t}}$. We simulate NIPROOF-SS($g, 2^{l(T+1-j)}, g^{R_{c_i}}, R_{c_i}^{2^{l(T+1-j)}}$) and run (t, n) -VSS PROCEDURE to share R_{c_i} , $1 \leq i \leq n-t$, with other dealers.
4. Finally, we compute $Y = \prod_{i=1}^t Y_{b_i} \prod_{j=1}^{n-t} Y_{c_j} \bmod N$ and sets $\mathcal{H}(j, Y, M) = \sigma$.

The above simulated view is identical to the real view. If the real view discloses any information, the adversary can simulate the real run to get information. Since the view of the simulator gives no useful information, the real view also provides no useful information.

Obtaining a forgery. Let c be the time period that \mathcal{F} switches to the BREAKIN phase. We provide the secret key S_c to \mathcal{F} and run \mathcal{F} to output a forged signature (c', Z, α) for M' , where $c' < c$. The (c', Z, α) is a forged signature for the single-user FS-DS. Since the single-user scheme FS-DS is secure, our distributed scheme TFSS is secure. This completes the proof. \square

5.4 Discussion

Proactive security. We can easily add the proactive mechanism to TFSS.UPDATE. The only difference is to compute $\hat{s}_{i,k} = 1/s_{i,k}^{2^{l(T-j)}}$ in step 2, instead of $\hat{s}_{i,k} = 1/s_{i,k}^{2^{l(T-j+1)}}$, and new secret-key share $S'_{i,j} = \prod_{k=1}^n s_{k,i} \bmod N$ in the refresh phase. Furthermore, $s_{i,k}$ can be encrypted and sent to dealer D_k using D_k 's public-key share $P_{k,j}$. This saves the private channel.

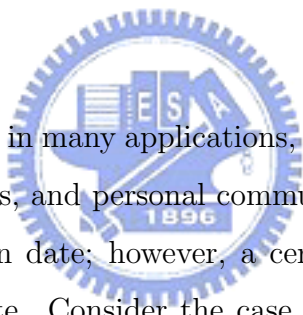
New construction. We can use polynomial secret sharing in our scheme, though it is less efficient. Our new construction is as follows. Initial setting is a bit different from that in Section 5.2. Let $f(x)$ be a degree- t polynomial with $f(0) = S_0$ and shared by all dealers by (t, n) -VSS PROCEDURE. To update the key S_j to S_{j+1} , all dealers compute the multiplication of two secrets for l times, where l is the security parameter. The robustness property is achieved by our SQ PROCEDURE. SQ PROCEDURE uses a proof to show that a dealer is honest. To compute a signature for a message, all dealers compute $l(T+1-j) + \log_2 \sigma$ times of distributed multiplication of secrets for $Y = R^{2^{l(T+1-j)}} \bmod N$ and $Z = RS_j^{\sigma} \bmod N$.

Efficiency. In our *new construction* based on polynomial secret sharing, dealers perform l multiplications of shares to update the key. That is, they exchange messages l times and compute l proofs for MULT PROCEDURE. To compute a signature, dealers exchange $l(T+1-j) + \log_2 \sigma$ messages and compute $l(T+1-j) + \log_2 \sigma$ proofs. As we can see, the computation and communication costs are quite expensive.

In *our main scheme* in Section 5.2, we combine the techniques of polynomial secret sharing and multiplicative secret sharing to reduce the cost. Each dealer exchanges messages twice in the key update stage, and once in the signing message stage. Each dealer needs to compute one proof in both key update and signing message stages. Therefore, our main scheme is quite efficient.

Chapter 6

Applications to key-evolving public key certificate-based authentication protocol



Certificates have been used in many applications, such as electronic commerce, accessing Internet resources, and personal communications services, etc. Certificates have an expiration date; however, a certificate may become invalid prior to the expiration date. Consider the case that a subscriber subscribes services from a service provider (SP). When a subscriber requests the services, SP has to identify the identity of the subscriber. Usually, the subscriber convinces SP of his secret key issued by SP. However, the secret key may have been compromised or lost. Once a subscriber knows that his secret key is disclosed, he informs SP to add his certificate to CRLs for securing the subscriber's right and protecting the system. After receiving the identified notification from a subscriber, SP adds subscriber's certificate to CRLs. If the lifetime of a certificate is two years, the certificate is listed in the CRLs two years at most. This is a quite cost for SP. Furthermore, a subscriber may not know that his secret key was compromised already. To save the cost of CRLs for SP and secure subscriber's right, we provide a solution to reduce these costs.

The detailed format of a certificate defined in X.509 [96] contains the user's

public key and other information and a signature signed by CA (Certificate Authority). For simplicity, we consider that a traditional public key certificate as follows.

$$\text{Cert}_{U_i} = \{ID_{U_i}, KU_{U_i}, \text{Date}_{U_i}, L_{U_i}, (ID_{U_i}, KU_{U_i}, \text{Date}_{U_i}, L_{U_i})_{KR_{SP}}\}$$

where Cert_{U_i} represents the certificate of U_i , in which ID_x means the identity of entity X, KU_x is the public key of entity X, Date_x is the issue date of the certificate to X, KR_x denotes the private key of entity X and L_x is the life time. The key-evolving public key certificate is a bit difference from the traditional public key certificate. The later one contains a base public key $KU_{U_i,0}$, the length of a time period I_t and the most large time period T .

$$\text{Cert}_{U_i} = \{ID_{U_i}, KU_{U_i,0}, \text{Date}_{U_i}, L_{U_i}, I_t, T, (ID_{U_i}, KU_{U_i,0}, \text{Date}_{U_i}, L_{U_i}, I_t, T,)_{KR_{SP}}\}$$

SP uses CRL to save the revoked certificates. Many strategies of certificate revocation have proposed in much literature. Storage cost and communication cost are two primary measures in the strategies of certificate revocation. Here, we concentrate on reducing the storage cost. To reduce the cost for saving revoked certificates, we introduce the key-evolving public key encryption scheme to the public key certificate-based protocols. We stress that only the public key encryption certificate-based protocols are considered. Such certificate-based protocols split time into time periods. Let the lifetime of public key certificate is divided into time periods, says T. At time period j , a subscriber convinces SP of his secret key SK_j and so does SP. At time period j , if an encryptor wants to send M to the subscriber, he computes a ciphertext $C = E(PK_j, M)$ of M , where PK_j is the public key at time period j . The subscriber uses SK_j of time period j to decrypt C and obtains M . When time makes a transit from j to $j + 1$, the secret key SK_j becomes invalid. It is worth to note that the key-evolving public key certificate is invariant during the lifetime of the certificate. As a result, the server need not frequently issue certificate and distribute certificate. Since the last secret key is automatically revoked in the new time period, CRLs of time period j need not be maintained at time period

- | |
|--|
| <ol style="list-style-type: none"> 1. $U \rightarrow SP: E_{PK_{S,j}}(c_1), G(c_1) \oplus Cert_U$ 2. $SP \rightarrow U: E_{PK_{U,j}}(c_1 c_2)$ 3. $U \rightarrow SP: c_2$ |
|--|

Figure 6.1: A key-evolving certificate-based authentication protocol

$j + 1$. At the beginning of each time period, CRLs are reset to zero so that the storage cost of CRLs is reduced. Additional cost for a subscriber is key update. A subscriber need to update his secret key SK_j to SK_{j+1} when time makes a transit from j to $j + 1$.

First, we describe a simple and concrete encryption certificate-based authentication protocol. Then, we extend to other applications. Furthermore, we discuss the relevant issues of implementation.

6.1 Key-evolving public key certificate-based authentication protocol

Assume that SP provides the various Internet services. When a user wishes to subscribe the services, he need register his identity to the SP. After registration, SP gives the subscriber two smart cards. One card saves a certificate, system's certificate and other information. The other saves all secret keys except initial secret key. The subscriber's certificate contains a key-evolving public key PK and other information. The current time period, says j , is known to the subscriber and SP. Assume that G is an pseudorandom number generator, which takes as input a seed and outputs random bits with enough length. Let U denote the subscriber, E the key-evolving encryption scheme, $Cert_U$ the certificate of U , $PK_{U,j}$ the public key of U and $PK_{S,j}$ the public key of SP at time period j . A simple and concrete authentication procedure as shown in Figure 6.1.

This is an encryption certificate-based protocol. First, a subscriber U sends his key-evolving public key certificate and a random number c_1 encrypted by $PK_{S,j}$ to SP as a request. SP checks if U is in CRL. If so, SP then rejects the request. Otherwise, SP selects a random number c_2 , and sends U the ciphertext of $(c_1||c_2)$ encrypted by $PK_{U,j}$ as a challenge. Finally, U decrypts the ciphertext by using SK_j and obtains $(c'_1||c'_2)$. If c'_1 is equal to c_1 , he returns c'_2 as the response. Otherwise, U stops the authentication procedure. After receiving the response c'_2 , SP checks if c'_2 is equal to c_2 . If so, SP allows U to access the resources. Otherwise, he rejects U's request. One can develop authentication protocols satisfying his requirements. Further discussion is as follows.

Key agreement. To protect the content of communication hereafter, SP and U needs to establish a common session key after running authentication procedure. The protocol provides the function of key agreement. The common session key may be $c_1 \oplus c_2$.

Key update. At the beginning of each time period, U needs to update his secret key by the UPD algorithm (described in Chapter 3). By the aid of the second smart card, U updates the secret key.

Reducing the storage cost of CRLs. For SP, key-evolving encryption certificate-based authentication protocols can reduce the storage cost of CRLs and no additional cost for certificate issue and distribution is added since

1. A disclosed secret key SK_j becomes invalid at time period $j + 1$ so that CRLs of time period j cannot be maintained at time period $j + 1$. Thus, the size of CRLs is reset to zero at the beginning of next new time period.
2. The key-evolving public key certificate is invariant during the lifetime of the certificate. This fact results in that SP does not issue a new certificate to the user and thus certificate update and distribution.

For extreme case, if the time period is very shorter, e.g. one day, then CRLs may be eliminated. Since time period is very short, the possibility of

the disclosure of secret key is very small so that SP cannot maintain CRLs. However, the subscriber needs more frequently to update his secret key.

6.2 Extension

We extend to other applications, such as, encryption certificate-based key exchange and electronic commerce. Unlike conventional public key certificate, we use the key-evolving public key certificate in these protocols. Thus, a subscriber and the server who own the secret key of the current time period can finish the transaction. Since an old secret key is invalid at new time period, the server merely needs to maintain CRLs of the current time period. Even a subscriber loses his current secret key, his right is protected at new time period. A thief obtaining the secret key SK_j of a subscriber can convince the server at time period j only.



6.3 Security

Since the encryption certificate-based authentication protocols use the key-evolving encryption scheme, the security is concerned. Assume that the original certificate-based protocols and the key-evolving public key encryption scheme are secure. The new certificate-based protocols are still secure since the new protocols merely use the key-evolving public key encryption scheme to replace the conventional public key encryption scheme. The security of the key-evolving encryption scheme is discussed in Chapter 3.

6.4 Implementation

We discuss relevant issues for implementing the encryption certificate-based authentication protocol.

Registration. When subscribing the service from SP, SP gives two smart cards to the subscriber. One card saves subscriber's certificate, the first secret key $f(1)$, and SP's certificate, etc. The other card saves other secret keys $\{SK_2, \dots, SK_T\}$, time period 1 and physical time. For a certificate with the lifetime of two years, the time period may be one month or two month and thus $T = 24$ or 12 . We can map the time $T=1$ to the current date, e.g. 29/07/2002, and thus the first secret key is $f(20020729)$. In addition, SP need record current time period and physical time of each subscriber.

Time synchronization. In the protocol, if the subscriber loses synchronization with SP, authentication protocol will fail. To avoid losing synchronization, before key updating, SP need send all subscriber a message consisting of the command of key updating, current time period, next time period and physical time. After receiving the command of key update, a subscriber can do key update. After finished, the subscriber send an acknowledge to SP. SP checks if the acknowledge is correct. If yes, SP updates subscriber's record of new time period and physical time to his database. If the subscriber cannot send an acknowledge to SP after three notifications of key updating, SP disables the service of the subscriber and contacts him by phone.

Key update. When updating key, via a computer, a smart card and related softwares (KUD algorithm), a subscriber can easily upgrade his secret key. The procedure of key update does not involve any trusted third party so that our schemes is very practical. The first card is carried outside and viewed as an on-line device. We stress that an on-line card is possibly carried to anywhere and used. Thus, the card is easily cracked by malicious adversaries and the secret key is disclosed. Another card that saves the information of key update is an off-line device. The off-line device is seldom used and merely used to upgrade the secret key at home . We view the card as a secure device.

Hash function. The scheme KEENCROM needs three hash functions. Currently, hash functions MD5 and SHA-1 [95] are used in a large variety of popular security applications and protocols such as TLS, SSL, PGP, SSH and

IPSec etc. However, MD5 can be found collisions in a powerful attack [94]. Wang and Yu call this kind of differential attack as *a modular differential* attack. Biham et al. [9] uses a generic mutli-block technique to find collisions. They use this technique to find a four-block collision of SHA-0 with complexity 2^{51} . Also, this technique allows us to find collisions of reduced versions of SHA-1. Wang et al. [95] show that collisions of SHA-1 can be found with complexity less than 2^{69} hash operations such that the full 80-step SHA-1 with complexity less than the 2^{80} theoretical bound. Although, the recent research on MD5 collision should have little impact on the use of MD5, MD5 is still secure against a brute force attack [90]. A collision of MD5 can only be produced using very specific input blocks. In the real world, these types of input blocks do not occur. So, you can use MD5, SHA-1 and the family of SHA-2 to implement hash functions. SHA-224, SHA-256, SHA-384, and SHA-512 that are referred to as SHA-2. You can choose appropriate one to implement H_1 , H_2 and H_3 . Collisions found in MD5 and SHA-1 do not affect the security of our scheme. You can recall the encryption algorithm that outputs a ciphertext $= \langle j, \alpha, \beta_1, \beta_2, s, h \rangle$, where $\alpha = g^k$, $\beta_1 = r \cdot (\prod_{i=0}^z (g^{a_i})^{j^i})^k = r \cdot g^{f(j) \cdot k}$, $\beta_2 = k \oplus H_1(j, r)$, $s = m \oplus H_2(j, r, k)$, $h = H_3(j, r, k, m)$. Give α , β_1 and β_2 , then r and k are fixed. At time period j , $H_1(j, r)$ and $H_2(j, r, k)$ are fixed such that m is fixed. So, $H_3(j, r, k, m)$ also is fixed. Even though you can find m' satisfying $H_3(j, r, k, m') = H_3(j, r, k, m)$, this does not affect the security of our scheme. The key point is that (j, r, k) are fixed and can be verified by checking α , β_1 and β_2 . So, an attacker cannot arbitrarily modify a ciphertext unless he knows (r, k, m) . Furthermore, an attacker cannot get more information except he has known already.



Chapter 7

Conclusion and Future Work

7.1 Conclusion

We have proposed three KE-ENC schemes to deal with the key exposure problem of public key cryptosystems. Our schemes are the first key-evolving public key encryption schemes. Our schemes achieve z -resilience so that compromise of z private keys does not affect confidentiality of messages encrypted in other time periods. In these schemes, a ciphertext has the concept of the time stamp. The basic scheme is semantically secure against passive adversaries under standard cryptographic assumptions. The modification of the basic scheme can achieve the adaptive chosen ciphertext attack under the random oracle model. The third scheme achieves the adaptive chosen ciphertext attack under the standard model. These schemes are ElGamal-like public key encryption schemes and very simple. Furthermore, our key-evolving encryption schemes have the property of the forward and backward securities. The decryptor evolves his secret key via the aid of TA's in a distributed way. Our encryption scheme can be applied to public key certificate-based encryption protocols to reduce the size of CRL and thus save the cost of storage for SP. Finally, we discuss time synchronization among SP and all subscribers.

In addition, we have proposed a distributed threshold forward-secure signature to enhance the security of Abdalla and Reyzin's forward-secure signature

scheme, which is based on the 2^l -th root problem. Our scheme is robust and efficient in terms of the number of rounds so that the amount of exchanged messages among dealers is low. We prove that our scheme has the property of forward secrecy as long as Abdalla and Reyzin's scheme is a forward-secure signature scheme.

7.2 Future work

We describe future works as follows.

1. *Removing the trusted server.* Our schemes need a trusted server or assume that a secure device exists for key updating procedure. If the security of the trusted server and secure device are not guaranteed, the security of our schemes is also not guaranteed. In practical, the existence of the trusted server will affect the efficiency of the our schemes. Further research is to remove the trusted mechanism.
2. *Beyond z -resilience.* Our schemes are limited to the z -resilience. Once an adversary obtains more than z keys, he can compute the master secret key with the obtained keys. Thus, our encryption schemes are insecure. Finding another style of a key-evolving encryption scheme that is not limited to the z -resilience is also another research direction.
3. *RSA-like key-evolving encryption schemes.* In this thesis, we only design the ElGamal-like public key encryption schemes. However, RSA is another very simple and popular public key cryptosystem. We can focus on another way to construct a new key-evolving encryption scheme. RSA-like key-evolving encryption schemes are another research direction.





Bibliography

- [1] M. Abdalla, J. An, M. Bellare, C. Namprempre, "From identification to signatures via the Fiat-Shamir transform: minimizing assumptions for security and forward-security." *Proceedings of Advances in Cryptology – Eurocrypt 2002*, Lecture Notes in Computer Science 1716, Springer-Verlag, 2002.
- [2] N. Alon, Z. Galil, M. Yung, "Dynamic-resharing verifiable secret sharing", *European Symposium on Algorithms 95 (ESA 95)*, Lecture Notes in Computer Science 979, pp.523-537, Springer-Verlag, 1995.
- [3] M. Abdalla, S. Miner, C. Namprempre, "Forward security in threshold signature schemes", *Topics in Cryptology – CT-RSA 2001*, Lecture Notes in Computer Science 2020, pp. 441–456, Springer-Verlag, 2001.
- [4] M. Abdalla, L. Reyzin, "A new forward-secure digital signature scheme", *Proceedings of Advances in Cryptology – Asiacrypt 2000*, Lecture Notes in Computer Science 1976, pp.116-129, Springer-Verlag, 2000.
- [5] C. Adams, P. Sylvester, M. Zolotarev, R. Zuccherato, "Internet X.509 public key infrastructure data validation and certification server protocols", *RFC 3029*, IETF PKIX Working Group, 2001.
- [6] R. Blakley, "Safeguarding cryptographic keys", *Proceedings of FIPS Conference*, pp.313-317, 1979.

- [7] D. Boneh, "The decision Diffie-Hellman problem", *Proceedings of the Third Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science 1423, pp.48-63, Springer-Verlag, 1998.
- [8] D. Boneh, X. Boyen, E.J. Goh, "Hierarchical identity based encryption with constant size ciphertext", *Proceedings of Advances in Cryptology – Eurocrypt 2005*, Lecture Notes in Computer Science 3493, pp. 440-456, Springer-Verlag, 2005.
- [9] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, W. Jalby, "Collisions of SHA-0 and Reduced SHA-1", *Proceedings of Advances in Cryptology - EUROCRYPT 2005*, Lecture Notes in Computer Science 3494, pp. 36-57, Springer-Verlag, 2005
- [10] D. Boneh, M. Franklin, "Identity based encryption from the Weil pairing", *Proceedings of Advances in Cryptology – Crypto 2001*, Lecture Notes in Computer Science 2139, pp.213-229, Springer-Verlag, 2001.
- [11] M. Ben-Or, S. Goldwasser, A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computations", *Proceedings of the 20th ACM Symposium on Theory of Computing*, pp.1-10, ACM, 1988.
- [12] M. Bellare, P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *Proceedings of the First ACM Conference on Computer and Communications Security*, pp.62-73, ACM, 1993.
- [13] M. Bellare, S.K. Miner, "A forward-secure digital signature scheme", *Proceedings of Advances in Cryptology – Crypto '99*, Lecture Notes in Computer Science 1666, pp.431-448, Springer-Verlag, 1999.
- [14] M. Bellare, A. Palacio, "Protecting against key exposure:strongly key-insulated encryption with optimal threshold", *Applicable Algebra in En-*

- gineering, Communication and Computing* Vol.16 , Issue 6, pp.379-396, 2006.
- [15] M. Bellare, B. Yee, "Forward security in private-key cryptography", *Topics in Cryptology – CT-RSA 03*, Lecture Notes in Computer Science 2612, Springer-Verlag, 2003.
- [16] D. Cooper, "A model of certificate revocation", *Proceedings of 15th Annual Computer Security Applications Conference*, pp. 256-264, 1999.
- [17] D. Chaum, C. Crepeau, I. Damgard, "Multiparty unconditionally secure protocols", *Proceedings of the 20th ACM Symposium on Theory of Computing*, pp.11-19, ACM, 1988.
- [18] R. Canetti, O. Goldreich, S. Halevi, "The random oracle methodology revisited", *Proceedings of the 30th ACM Annual Symposium on Theory of Computing*, pp.209-218, ACM, 1998.
- [19] R. Canetti, A. Herzberg, "Maintaining security in the presence of transient faults", *Proceedings of Advances in Cryptology – Crypto '94*, Lecture Notes in Computer Science 839, pp. 425-438, Springer-Verlag, 1994.
- [20] R. Canetti, R. Gennaro, A. Herzberg, D. Naor, "Proactive security: long-term protection against break-ins", *CryptoBytes* 3(1), 1997.
- [21] R. Canetti, S. Halevi, J. Katz, "A forward-secure public-key encryption scheme", *Proceedings of Advances in Cryptology – Eurocrypt 2003*, Lecture Notes in Computer Science 2656, pp. 255-271, Springer-Verlag, 2003.
- [22] D. Chaum, T. Pedersen, "Wallet databases with observers", *Proceedings of Advances in Cryptology – Crypto '92*, Lecture Notes in Computer Science 740, pp.89-105, Springer-Verlag, 1992.
- [23] R. Cramer, V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", *Proceedings of Advances*

- in Cryptology – Crypto '98*, Lecture Notes in Computer Science 1462, pp.13-25, Springer-Verlag, 1998.
- [24] Y. Desmedt, Y. Frankel, "Threshold cryptosystems", *Proceedings of Advances in Cryptology – Crypto '89*, Lecture Notes in Computer Science 435, pp.307-315, Springer-Verlag, 1989.
- [25] Y. Dodis, J. Katz, S. Xu, M. Yung, "Key-insulated public-key cryptosystems", *Proceedings of Advances in Cryptology – Eurocrypt 2002*, Lecture Notes in Computer Science 2332, pp.65-82, Springer-Verlag, 2002.
- [26] Y. Dodis, J. Katz, S. Xu, M. Yung, "Strong Key-insulated signature schemes", *Proceedings of 6th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2003*, Lecture Notes in Computer Science 2567, pp.130-144, Springer-Verlag, 2003.
- [27] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory* 31(4), pp.469-472, IEEE, 1985.
- [28] J. Elson, D. Estrin, "Time synchronization for wireless sensor networks", *Proceedings of the 15th International Parallel and Distributed Processing Symposium(IPDPS-01)*, IEEE Computer Society, 2001.
- [29] J.E. Elson, L. Girod, D. Estrin, "Fine-grained network time synchronization using reference broadcasts", *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation(OSDI)*, pp.147-163, 2002.
- [30] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing", *Proceedings of the 28th IEEE Annual Symposium on the Foundations of Computer Science*, pp.427-437, IEEE, 1987.
- [31] U. Feige, A. Fiat, A. Shamir, "Zero-knowledge proof of identity", *Journal of Cryptology*, Vol.1, No.2, pp.77-94, Springer-Verlag, 1988.

- [32] Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung, "Optimal-resilience proactive public-key cryptosystems", *Proceedings of 38th Annual Symposium on Foundations of Computer Science*, pp.384-393, IEEE, 1997.
- [33] Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung, "Proactive RSA", *Proceedings of Advances in Cryptology – Crypto '97*, Lecture Notes in Computer Science 1294, pp.440-454, Springer-Verlag, 1997.
- [34] B. Fox, B. LaMacchia, "Certificate Revocation: Mechanics and Meaning", *Proceedings of the Second International Conference on Financial Cryptography*, Lecture Notes in Computer Science 1465, pp.158-164, Springer-Verlag, 1998.
- [35] Y. Frankel, P. MacKenzie, M. Yung, "Adaptively-secure optimal-resilience proactive RSA", *Proceedings of Advances in Cryptology – Asiacrypt '99*, Lecture Notes in Computer Science 1716, pp.180-194, Springer-Verlag, 1999.
- [36] A. Fiat, A. Shamir, "How to prove yourself: practical solutions to identification and signature problems", *Proceedings of Advances in Cryptology – Crypto '86*, Lecture Notes in Computer Science 263, pp.186-194, Springer-Verlag, 1986.
- [37] P. Gemmel, "An introduction to threshold cryptography", *CryptoBytes* Vol.2, No.7, 1997.
- [38] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust threshold DSS signatures", *Proceedings of Advances in Cryptology – Eurocrypt '96*, Lecture Notes in Computer Science 1070, pp.354-371, Springer-Verlag, 1996.
- [39] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust and efficient sharing of RSA functions", *Proceedings of Advances in Cryptology – Crypto '96*, Lecture Notes in Computer Science 1109, pp.157-172, Springer-Verlag, 1996.

- [40] S. Ganeriwal, R. Kumar, M.B. Srivastava, "Timing-sync protocol for sensor networks", *Proceedings of the First ACM Conference on Embedded Networked Sensor System(SenSys)*, pp. 138-149, ACM, 2003.
- [41] S. Goldwasser, S. Micali, "Probabilistic encryption", *Journal of Computer and System Sciences* 28, pp.270-299, 1984.
- [42] L.C. Guillou, J. Quisquater, "A "paradoxical" identity-based signature scheme resulting from zero-knowledge", *Proceedings of Advances in Cryptology – Crypto '88*, Lecture Notes in Computer Science 403, pp.216-231, Springer-Verlag, 1988.
- [43] L. Guillou, J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory", *Proceedings of Advances in Cryptology – Eurocrypt '88*, Lecture Notes in Computer Science 330, pp.123-128, Springer-Verlag, 1988.
- [44] R. Gennaro, M. Rabin, T. Rabin, "Simplified VSS and fast-track multi-party computations with applications to threshold cryptography", *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing (PODC)*, pp.101 - 111, ACM, 1998.
- [45] C. Gentry, A. Silverberg, "Hierarchical ID-based cryptography", *Proceedings of Advances in Cryptology – Asiacrypt 2002*, Lecture Notes in Computer Science 2501, pp.548-566, Springer-Verlag, 2002.
- [46] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, "Proactive public key and signature systems", *Proceedings of the 4th ACM Symposium on Computer and Communication Security*, pp.100-110, ACM, 1997.
- [47] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive secret sharing or: how to cope with perpetual leakage", *Proceedings of Advances in Cryptology – Crypto '95*, Lecture Notes in Computer Science 963, pp.339-352, Springer-Verlag, 1995.

- [48] J. Horwitz, B. Lynn, "Toward hierarchical identity-based encryption." *Proceedings of Advances in Cryptology – Eurocrypt 2002*, Lecture Notes in Computer Science 2332, pp. 466-481, Springer-Verlag, 2002.
- [49] G. Itkis, L. Reyzin, "Forward-Secure Signatures with optimal signing and verifying", *Proceedings of Advances in Cryptology – Crypto 2001*, Lecture Notes in Computer Science 2139, pp. 332-354, Springer-Verlag, 2001.
- [50] G. Itkis, L. Reyzin, "SiBIR:signer-base intrusion-resilient signatures", *Proceedings of Advances in Cryptology – Crypto 2002*, Lecture Notes in Computer Science 2442, pp. 18-22, Springer-Verlag, 2002.
- [51] I. Ingemarsson, G.J. Simmons, "A protocol to set up shared secret schemes without the assistance of a mutually trusted party", *Proceedings of Advances in Cryptology – Eurocrypt '90*, Lecture Notes in Computer Science 473, pp.266-282, Springer-Verlag, 1990.
- [52] G. Itkis, P. Xie, "Generalized key-evolving signature schemes or how to foil an armed adversary", *The First MiAn International Conference on Applied Cryptography and Network Security*, Lecture Notes in Computer Science 2846, pp. 151 - 168, Springer-Verlag, 2003.
- [53] P. Kocher, "On certificate revocation and validation", *Proceedings of the Second International Conference on Financial Cryptography*, Lecture Notes in Computer Science 1465, pp. 172-177, Springer-Verlag, 1998.
- [54] H. Krawczyk, "Simple forward-secure signatures from any signature scheme", *Proceedings of the Seventh ACM Conference on Computer and Communication Security*, pp. 108-115, ACM, 2000.
- [55] J. Kim, K. Kim, "Intrusion-resilient key-evolving schnorr signature", *Proceedings of Computer Security Symposium 2003*, pp.379-384, 2003.

- [56] H. Kopetz and W. Ochsenreiter, "Clock synchroniziation in distributed real-time systems", *IEEE Transactions on Computers*, C-36(8), pp.933-939, IEEE, 1987.
- [57] L. Lamport, "Time, clocks, and the ordering of events in a distributed system", *Communications of the ACM*, 21(4), pp.558-565, ACM, 1978.
- [58] L. Lamport, P.M. Melliar-Smith, "Synchronizing clocks in the presence of faults", *Journal of the ACM*, 32(1), ACM, 1985.
- [59] C.F. Lu, S.W. Shieh, "Secure key-evolving protocols for discrete logarithm schemes." *Proceedings of CT-RSA 02*, pp.300-310, 2002.
- [60] Z. Le, Y. Ouyang, J. Ford, F. Makedon, "A hierarchical key-insulated signature scheme in the CA trust model", *Proceedings of Information Security (ISC 2004)*, Lecture Notes in Computer Science 3225, pp. 280-291. Springer-Verlag, 2004.
- [61] F. Mattern, "Virtual time and global states of distributed systems", *Proceedings of the International Workshop on Parallel and Distributed Algorithms*, 1988.
- [62] D.L. Mills, "Internet time synchronization: the network time protocol", *IEEE Transactions on Communications COM 39* No.10, pp.1482-1493, IEEE, 1991.
- [63] D.L. Mills, "Network time protocol (version 3) specification, implementation and analysis", *RFC 1305*, IETF Network Working Group, 1992.
- [64] S. Micali, "A secure and efficient digital signature algorithm," *Technical Report MIT/LCS/TM-501*, Massachusetts Institute of Technology, Cambridge, MA, 1994.
- [65] D.L. Mills, "Simple network time protocol (SNTP)", *RFC 1769*, IETF Network Working Group, 1995.

- [66] S. Micali, "Efficient certificate revocation", *Technical Report TM-542b*, MIT Laboratory for Computer Science, 1996.
- [67] M. Myers, "Revocation: options and challenges", *Proceedings of the Second International Conference on Financial Cryptography*, Lecture Notes in Computer Science 1465, pp. 165-171, Springer-Verlag, 1998.
- [68] P. McDaniel, S. Jamin, "Windowed certificate revocation", *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE Infocom 2000*, pp. 1406-1414, IEEE, 2000.
- [69] T. Malkin, D. Micciancio, S. Miner, "Efficient generic forward-secure signatures with an unbounded number of time periods." *Proceedings of Advances in Cryptology – Eurocrypt 2002*, Lecture Notes in Computer Science 2332, pp. 400 - 417, Springer-Verlag, 2002.
- [70] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, "The flooding time synchronization protocol", *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, pp. 39-49, ACM, 2004.
- [71] P. McDaniel, A. Rubin, "A response to 'Can we eliminate certificate revocation lists?'" , *Proceedings of the 4th International Conference on Financial Cryptography* Lecture Notes In Computer Science 1962, pp. 245 - 258, Springer-Verlag, 2000.
- [72] U.M. Maurer, Y. Yacobi, "A non-interactive public-key distribution system", *Designs, Codes and Cryptography*, Vol. 9, No.3, pp.305-316, 1996.
- [73] M. Naor, K. Nissim, "Certificate revocation and certificate update", *Proceedings of 7th USENIX Security Symposium*, pp. 217-228, 1998.
- [74] Nicolás G.D., Olivier M., Emmanuel D., "A new key-insulated signature scheme", *Proceedings of Sixth International Conference on Information*

and Communications Security Lecture Notes In Computer Science 3269, Springer-Verlag, 2004.

- [75] H. Ong, C. Schnorr, "Fast signature generation with a Fiat-Shamir like scheme", *Proceedings of Advances in Cryptology – Eurocrypt '90*, Lecture Notes in Computer Science 473, pp.432-440, Springer-Verlag, 1990.
- [76] K. Ohta, T. Okamoto, "A modification of the Fiat-Shamir scheme", *Proceedings of Advances in Cryptology – Crypto '88*, Lecture Notes in Computer Science 403, pp.232-243, Springer-Verlag, 1988.
- [77] R. Ostrovsky, M. Yung, "How to withstand mobile virus attacks", *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 51-61, ACM, 1991.
- [78] J. Postel, "Daytime protocol", *RFC 867*, IETF Network Working Group, 1983.
- [79] J. Postel, K. Harrenstien, "Time protocol", *RFC 868*, IETF Network Working Group, 1983.
- [80] T. Pedersen, "A threshold cryptosystem without a trusted party", *Proceedings of Advances in Cryptology – Eurocrypt '91*, Lecture Notes in Computer Science 547, pp.522-526, Springer-Verlag, 1991.
- [81] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing", *Proceedings of Advances in Cryptology – Crypto '91*, Lecture Notes in Computer Science 576, pp.129-140, Springer-Verlag, 1991.
- [82] R. Rivest, "Can we eliminate certificate revocation lists?", *Proceedings of the Second International Conference on Financial Cryptography*, Lecture Notes in Computer Science 1465, pp. 178-183, Springer-Verlag, 1998.
- [83] K. Römer, "Time synchronization in ad hoc networks", *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, ACM, 2001.

- [84] P. Ramanathan, K.G. Shin, R.W. Butler, "Fault-tolerant clock synchronization in distributed systems", *IEEE Computer*, vol.23, no.10, pp.33-42, IEEE, 1990.
- [85] T. Rabin, "A simplified approach to threshold and proactive RSA", *Proceedings of Advances in Cryptology – Crypto '98*, Lecture Notes in Computer Science 1462, pp.89-104, Springer-Verlag, 1998.
- [86] C. Rackoff, D. Simon, "Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack", *Proceedings of Advances in Cryptology – Crypto '91*, Lecture Notes in Computer Science 576, pp.433-444, Springer-Verlag, 1991.
- [87] A. Shamir, "Identity-based cryptosystems and signature schemes", *Proceedings of Advances in Cryptology – Crypto '84*, Lecture Notes in Computer Science 196, pp.47-53, Springer-Verlag, 1984.
- [88] A. Shamir, "How to share a secret", *Communications of the ACM* 22(11), pp.612-613, ACM, 1979.
- [89] B. Simons, J. Welch, N. Lynch, "An overview of clock synchronization", *Technical Report RJ 6505*, IBM Almaden Research Center, 1988.
- [90] E. Thompson, "MD5 collisions and the impact on computer forensics" *Digital Investigation*, Vol.2, No.1, pp. 36-40, 2005.
- [91] R.N. Wright, P.D. Lincoln, J.K. Millen, "Efficient fault-tolerant certificate revocation", *Proceedings of ACM Conference on Computer and Communications Security '00*, pp.19-24, ACM, 2000.
- [92] X. Wang, X. Lai, D. Feng, H. Chen, X. Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD", *Proceedings of Advances in Cryptology - EUROCRYPT 2005*, Lecture Notes in Computer Science 3494, pp. 1-18, Springer-Verlag, 2005

- [93] H. Wang, L. Yip, D. Maniezzo, J.C. Chen, R.E. Hudson, J. Elson, K. Yao, "A wireless time-synchronized COTS sensor platform part II-applications to beamforming", *Proceedings of IEEE CAS Workshop on Wireless Communications and Networking*, IEEE, 2002.
- [94] X. Wang, H. Yu, "How to Break MD5 and Other Hash Functions", *Proceedings of Advances in Cryptology - EUROCRYPT 2005*, Lecture Notes in Computer Science 3494, pp. 19-35, Springer-Verlag, 2005
- [95] X. Wang, Y.L. Yin, H. Yu, "Finding Collisions in the Full SHA-1", *Proceedings of Advances in Cryptology - CRYPTO 2005*, Lecture Notes in Computer Science 3621, pp.17-36, Springer-Verlag, 2005.
- [96] CCITT Recommendation X.509, "The Directory-Authentication Framework", 1993.
- [97] D. Yao, N. Fazio, Y. Dodis, A. Lysyanskaya, "ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption", *Proceedings of the 11th ACM Conference on Computer and Communications Security '04*, pp. 354-363, ACM, 2004.

Appendix A

In the following, we review well-known forward secure signature and encryption schemes. These schemes contain Bellare and Miner's, Abdalla and Reyzin's, Itkis and Reyzin's and Canetti et al's scheme.

Bellare and Miner's forward-secure signature scheme



Bellare and Miner proposed the first forward-secure signature scheme based on the difficulty of computing the square roots modulus a Blum integer [13]. Their scheme consists of four procedures: key generation (KG), key update (KUD), signing (SGN) and verification (VF). Let k and l be security parameters and T the largest time period. The scheme is shown in Figure 7.1 and 7.2. Initially, a signer randomly selects l secret keys over Z_N^* as the initial secret key SK_0 . Then, he computes the public key PK via the l secret keys and T . When the time transits from $j - 1$ to j , the signer computes the secret key $S_{i,j} = S_{i,j-1}^2 \pmod N$ for $i = 1, \dots, l$. To sign a message M , the signer selects a random R and computes $Y = R^{2^{(T+1)}} \pmod N$, $c_1 \cdots c_l = H(j, Y, M)$ and $Z = R \prod_{i=1}^l S_{i,j}^{c_i} \pmod N$. The signature of M is $\langle j, (Y, Z) \rangle$. To verify a signature $\langle j, (Y, Z) \rangle$ of M , the verifier computes $c_1 \cdots c_l = H(j, Y, M)$. and checks if $Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^l U_i^{c_i} \pmod N$.

-
1. $\text{KG}(k, l, T)$
 - (a) Pick random, distinct $k/2$ primes p, q such that $p \equiv q \equiv 3 \pmod{4}$.
 - (b) Let $N = pq$.
 - (c) Select $S_{1,0}, S_{2,0}, \dots, S_{l,0} \in Z_N^*$ at random.
 - (d) Set $U_i = S_{i,0}^{2^{(T+1)}} \pmod{N}$ for $i = 1, \dots, l$.
 - (e) Let $SK_0 = (N, T, 0, S_{1,0}, S_{2,0}, \dots, S_{l,0})$ and $PK = (N, T, U_1, \dots, U_l)$.

 2. $\text{KUD}(SK_{j-1})$ where $SK_{j-1} = (N, T, j-1, S_{1,j-1}, S_{2,j-1}, \dots, S_{l,j-1})$ and $1 \leq j \leq T+1$.
 - (a) Set $S_{i,j} = S_{i,j-1}^2 \pmod{N}$ for $i = 1, \dots, l$.
 - (b) Let $SK_j = (N, T, j, S_{1,j}, S_{2,j}, \dots, S_{l,j})$
-

Figure 7.1: Bellare and Miner's forward-secure signature scheme is based on the hardness of the square root problem. (part 1)

Abdalla and Reyzin's forward-secure signature scheme

Abdalla and Reyzin proposed an improvement of Bellare and Miner's with a shorter key [4]. The security of their scheme is based the hardness of computing the 2^l -th root problem. Their scheme shown in Figure 7.3 also contains four procedures: KG, KUD, SGN and VF. Initially, the signer randomly selects S_0 over Z_N^* . Then, he sets the signing key $SK_0 = (N, T, 0, S_0)$ and computes the public key $PK = (N, U, T)$, where $U = 1/S_0^{2^{(T+1)}} \pmod{N}$. For any time period j , the secret key $SK_j = (N, T, j, S_{j-1}^{2^l})$, i.e., $S_j = S_{j-1}^{2^l}$. To sign a message M , the signer randomly selects $R \in Z_N^*$ and computes (j, Z, σ) , where $Z = RS_j^\sigma$

-
1. $\text{SGN}(H, SK_j, M)$ where $SK_j = (N, T, j, S_{1,j}, S_{2,j}, \dots, S_{l,j})$ and $H : 0, 1^* \rightarrow 0, 1^l$ is a random-oracle hash-function.
 - (a) Select $R \in Z_N^*$ at random.
 - (b) Set $Y = R^{2^{(T+1)}} \pmod N$.
 - (c) Let $c_1 \cdots c_l = H(j, Y, M)$.
 - (d) Compute $Z = R \prod_{i=1}^l S_{i,j}^{c_i} \pmod N$.
 - (e) Return $\langle j, (Y, Z) \rangle$.

 2. $\text{VF}(H, PK, M, \langle j, (Y, Z) \rangle)$ where $PK = (N, T, U_1, \dots, U_l)$.
 - (a) Compute $c_1 \cdots c_l = H(j, Y, M)$.
 - (b) Check if $Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^l U_i^{c_i} \pmod N$. If so, return 1. Otherwise, return 0.
-

Figure 7.2: Bellare and Miner's forward-secure signature scheme is based on the hardness of the square root problem. (part 2)

$\pmod N$, $\sigma = H(j, Y, M)$ and $Y = R^{2^{(T+1-j)}} \pmod N$. To verify a signature (j, Z, σ) of M , a verifier computes $Y' = Z^{2^{(T+1-j)}} U^\sigma \pmod N$ and checks if $\sigma = H(j, Y', M)$.

Itkis and Reyzin's forward-secure signature scheme

Itkis and Reyzin proposed a forward-secure signature scheme based on the GQ signature scheme [49]. The security of the scheme is based on e -th root problem. This scheme optimizes the procedures of signing and verifying. Their scheme also consists of four procedure: IR.KG, IR.KUD, IR.SGN and IR.VF.

The scheme is shown in Figure 7.4 and 7.5. In the beginning, the signer selects $t_1 \in Z_N^*$ and uses the *seed* to generate T primes e_1, \dots, e_T . Then, he sets $SK_1 = (1, T, N, s_1, t_2, e_1, \text{seed})$ and $PK = (n, v, T)$ as the first secret key and the public key, where $s_1 = t_1^{f_2} \pmod N$, $t_2 = t_1^{e_1} \pmod N$, $v = 1/s_1^{e_1} \pmod N$ and $f_2 = e_2 \cdots e_T \pmod{\phi(N)}$. When the time transits from j to $j + 1$, the secret key is updated to be $SK_{j+1} = (j + 1, T, N, s_{j+1}, t_{j+2}, e_{j+1}, \text{seed})$, where $s_{j+1} = t_{j+1}^{e_{j+2} \cdots e_T} \pmod N$ and $t_{j+2} = t_{j+1}^{e_{j+1}} \pmod N$. Remark that the *seed* is used regenerate T prime e_i . To sign a message M , the signer randomly selects R and computes the signature (z, σ, j, e_j) of M , where $z = rs_j^\sigma \pmod N$ and $\sigma = H(j, e_j, y, M)$ in which $y = r^{e_j} \pmod N$. To verify a signature (z, σ, j, e_j) of M , a verifier checks if $e \geq 2^l(1 + j/T)$ or $e < 2^l$ or e is even and $\sigma = H(j, e, y', M)$ in which $y' = z^e v \sigma \pmod N$.

Canetti et al.'s forward-secure public-key encryption scheme



Canetti et al proposed forward-secure public-key encryption scheme based on bilinear Diffie-Hellman assumption. They proposed forward-secure encryption schemes from any BTE(binary tree encryption) scheme. The BTE scheme is shown in Figure 7.6 and 7.7. Let ℓ denote the depth of the tree and $\omega|i = \omega_1 \cdots \omega_i$. In the scheme, they use a $(2\ell + 1)$ -wise independent family \mathcal{H} of functions $H : \{0, 1\}^{\leq \ell} \rightarrow \mathbf{G}_1$. Given elements $x_1, \dots, x_k \in \{0, 1\}^{\leq \ell}$ and $g_1, \dots, g_k \in \mathbf{G}_1$ (with $k \leq 2\ell + 1$), it is possible to efficiently sample a random $H \in \mathcal{H}$ satisfying $H(x_i) = g_i$ for $i = 1, \dots, k$.

To construct a forward-secure scheme with $N = 2^{\ell+1} - 1$ time periods, we simply use a BTE of depth ℓ and associate the time period with all nodes of the tree according to a pre-order traversal. The public key is simply the root public key for the BTE scheme. The private key for period i consists of the secret key for node ω^i as well as those for all right siblings of the nodes on the path from the root to ω^i . To encrypt a message at time period i , the message

is encrypted for node ω^i using the BTE scheme. The ciphertext at time period i is simply decrypted by the secret key of ω^i . The secret key is updated in the following: if ω^i is an internal node, then the secret keys for ω^{i+1} and its sibling are derived; otherwise the secret key for node ω^{i+1} is already stored as part of the secret key. In either case, the key for node ω^i is then deleted.



-
1. $\text{KG}(k, l, T)$.
 - (a) Select two large primes p and q such that $p \equiv q \equiv 3 \pmod{4}$, $2^{k-1} \leq (p-1)(q-1)$, and $pq < 2^k$. Let $N = pq$.
 - (b) Randomly select S_0 from Z_N^* and compute $U = 1/S_0^{2^{l(T+1)}} \pmod{N}$.
 - (c) Set $SK_0 = (N, T, 0, S_0)$ and $PK = (N, U, T)$.

 2. $\text{KUD}(SK_j)$ where $SK_j = (N, T, j, S_j)$.
 - (a) If $j = T$, set $SK_j = \text{null}$; otherwise, set $SK_{j+1} = (N, T, j + 1, S_j^{2^l} \pmod{N})$.

 3. $\text{SGN}(H, M, SK_j)$ where $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ a collision-resistant hash function.
 - (a) Randomly select $R \in Z_N^*$ and compute $Y = R^{2^{l(T+1-j)}} \pmod{N}$, $\sigma = H(j, Y, M)$, and $Z = RS_j^\sigma \pmod{N}$.
 - (b) The signature is (j, Z, σ) .

 4. $\text{VF}(H, PK, (j, Z, \sigma), M)$ where $PK = (N, U, T)$.
 - (a) If $Z \equiv 0$, return 0; otherwise, compute $Y' = Z^{2^{l(T+1-j)}} U^\sigma \pmod{N}$.
 - (b) Output 1 if and only if $\sigma = H(j, Y', M)$.
-

Figure 7.3: Abdalla and Reyzin's forward-secure signature scheme is based the hardness of the 2^l -th root problem.

1. IR.KG(k, l, T).

- (a) Generate random $\lceil k/2 \rceil$ bits primes q_1, q_2 such that $p_i = 2q_i + 1$ are primes.
- (b) Let $N = p_1 p_2$.
- (c) Select $t_1 \in Z_N^*$ at random.
- (d) Generate primes e_i using *seed* such that $2^l(1 + (i - 1)/T) \leq e_i < 2^l(1 + i/T)$ for $i = 1, 2, \dots, T$.
- (e) Compute $f_2 = e_2 \cdot \dots \cdot e_T \pmod{\phi(N)}$, $s_1 = t_1^{f_2} \pmod{N}$, $v = 1/s_1^{e_1} \pmod{N}$ and $t_2 = t_1^{e_1} \pmod{N}$.
- (f) Set $SK_1 = (1, T, N, s_1, t_2, e_1, \text{seed})$ and $PK = (n, v, T)$.

2. IR.KUD(SK_j) where $SK_j = (j, T, N, s_j, t_{j+1}, e_j, \text{seed})$.

- (a) If $j = T$ then return ϵ .
- (b) Regenerate e_{j+1}, \dots, e_T using *seed*.
- (c) Compute $s_{j+1} = t_{j+1}^{e_{j+2} \dots e_T} \pmod{N}$ and $t_{j+2} = t_{j+1}^{e_{j+1}} \pmod{N}$.
- (d) Set $SK_{j+1} = (j + 1, T, N, s_{j+1}, t_{j+2}, e_{j+1}, \text{seed})$.

Figure 7.4: Itkis and Reyzin's forward-secure signature scheme is based on GQ signature.(part 1)

-
1. IR.SGN(M, H, SK_j) where $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ a random oracle hash function.
 - (a) Select $r \in Z_N^*$ at random.
 - (b) Compute $y = r^{e_j} \pmod N$, $\sigma = H(j, e_j, y, M)$ and $z = rs_j^\sigma \pmod N$.
 - (c) Return (z, σ, j, e_j) .
 2. IR.VF($H, PK, M, (z, \sigma, j, e)$), where $PK = (N, v, T)$.
 - (a) If $e \geq 2^l(1 + j/T)$ or $e < 2^l$ or e is even then return 0.
 - (b) If $z \equiv 0 \pmod N$ then return 0.
 - (c) Compute $y' = z^e v \sigma \pmod N$.
 - (d) If $\sigma = H(j, e, y', M)$ then return 1 else return 0.
-

Figure 7.5: Itkis and Reyzin's forward-secure signature scheme is based on GQ signature.(part 2)

-
1. $\text{GEN}(1^k, \ell)$
 - (a) Run $IG(1^k)$ to generate groups $\mathbf{G}_1, \mathbf{G}_2$ with prime order q and bilinear map \hat{e} .
 - (b) Randomly select a generator $P \in G_1$ and $\alpha \in \mathbf{Z}_q$.
 - (c) Set $Q = \alpha P$.
 - (d) Choose a random function $H \in \mathcal{H}$.
 - (e) The public key is $PK = (\mathbf{G}_1, \mathbf{G}_2, \hat{e}, P, Q, \ell, H)$.
 - (f) The root secret key is $SK_\varepsilon = \alpha H(\varepsilon)$.
 2. $\text{DER}(PK, \omega, SK_\omega)$
 - (a) Let $\omega = \omega_1 \cdots \omega_t$.
 - (b) Parse SK_ω as $(R_{\omega|1}, R_{\omega|2}, \dots, R_{\omega|t-1}, R_\omega, S_\omega)$.
 - (c) Randomly select $\rho_{\omega 0}, \rho_{\omega 1} \in \mathbf{Z}_q$. Set $R_{\omega 0} = \rho_{\omega 0} P$, $R_{\omega 1} = \rho_{\omega 1} P$, $S_{\omega 0} = S_\omega + \rho_{\omega 0} H(\omega 0)$, and $S_{\omega 1} = S_\omega + \rho_{\omega 1} H(\omega 1)$.
 - (d) Output $SK_{\omega 0} = (R_{\omega|1}, R_{\omega|2}, \dots, R_{\omega|t-1}, R_{\omega 0}, S_{\omega 0})$ and $SK_{\omega 1} = (R_{\omega|1}, R_{\omega|2}, \dots, R_{\omega|t-1}, R_{\omega 1}, S_{\omega 1})$.

Figure 7.6: Canetti et al's binary tree encryption scheme is based on bilinear Diffie-Hellman assumption.(part 1)

1. $\text{ENC}(PK, \omega, M)$

(a) Let $\omega = \omega_1 \cdots \omega_t$. Select a random number $\gamma \in \mathbf{Z}_q$.

(b) Output $C = (\gamma P, \gamma H(\omega|1), \gamma H(\omega|2), \dots, \gamma H(\omega), M \cdot d)$, where $d = \hat{e}(Q, H(\varepsilon))^\gamma$.

2. $\text{DEC}(PK, \omega, SK_\omega, C)$

(a) Let $\omega = \omega_1 \cdots \omega_t$, parse SK_ω as $(R_{\omega|1}, \dots, R_\omega, S_\omega)$, and parse C as $(U_0, U_1, \dots, U_t, V)$.

(b) Output $M = V/d$, where $d = \frac{\hat{e}(U_0, S_\omega)}{\prod_{i=1}^t \hat{e}(R_{\omega|i}, U_i)}$.

Figure 7.7: Canetti et al's binary tree encryption scheme is based on bilinear Diffie-Hellman assumption.(part 2)