FLEXIBLE DIGITAL AUTHENTICATION TECHNIQUES

He Ge, M.S.

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

May 2006

APPROVED:

Stephen R. Tate, Major Professor
Roy T. Jacob, Committee Member
Ram Dantu, Committee Member
Krishna Kavi, Chair of the Department of
Computer Science and Engineering
Oscar Garcia, Dean of the College of
Engineering
Sandra L. Terrell, Dean of the Robert B.
Toulouse School of Graduate Studies

Ge, He. *Flexible Digital Authentication Techniques*. Doctor of Philosophy (Computer Science), May 2006, 105 pp., 1 table, 1 figure, references, 60 titles.

This dissertation investigates authentication techniques in some emerging areas. Specifically, authentication schemes have been proposed that are well-suited for embedded systems, and privacy-respecting pay Web sites.

With embedded systems, a person could own several devices which are capable of communication and interaction, but these devices use embedded processors whose computational capabilities are limited as compared to desktop computers. Examples of this scenario include entertainment devices or appliances owned by a consumer, multiple control and sensor systems in an automobile or airplane, and environmental controls in a building. An efficient public key cryptosystem has been devised, which provides a complete solution to an embedded system, including protocols for authentication, authenticated key exchange, encryption, and revocation. The new construction is especially suitable for the devices with constrained computing capabilities and resources. Compared with other available authentication schemes, such as X.509, identity-based encryption, etc, the new construction provides unique features such as simplicity, efficiency, forward secrecy, and an efficient re-keying mechanism.

In the application scenario for a pay Web site, users may be sensitive about their privacy, and do not wish their behaviors to be tracked by Web sites. Thus, an anonymous authentication scheme is desirable in this case. That is, a user can prove his/her authenticity without revealing his/her identity. On the other hand, the Web site owner would like to prevent a bunch of users from sharing a single subscription while hiding behind user anonymity. The Web site should be able to detect these possible malicious behaviors, and exclude corrupted users from future service. This dissertation extensively discusses anonymous authentication techniques, such as group signature, direct anonymous attestation, and traceable signature. Three anonymous authentication schemes have been proposed, which include a group signature scheme with signature claiming and variable linkability, a scheme for direct anonymous attestation in trusted computing platforms with sign and verify protocols nearly seven times more efficient than the current solution, and a state-of-the-art traceable signature scheme with support for variable anonymity. These three schemes greatly advance research in the area of anonymous authentication.

The authentication techniques presented in this dissertation are based on common mathematical and cryptographical foundations, sharing similar security assumptions. We call them flexible digital authentication schemes.

ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor, Dr. Stephen R. Tate. It was not always easy in the past three years for my Ph.D. study. Mainly through interaction with Dr. Tate, numerous problems were solved, and I shaped my understanding of my subject. Finally, a very basic idea has been evolved into some interesting, and exciting results presented in this dissertation. I am grateful to him for his valuable guidance, for being patience, and kind, and financial support. From Dr. Tate, I learned a tremendous amount about doing research and presenting it.

Thanks Dr. Roy T. Jacob, and Dr. Ram Dantu, for being my committee members, and their encouragement.

I deeply thank our department for offering me teaching assistant in the past five years to support my study. Without these financial assistant, it would be extremely difficult to continue my research work.

Last, but certainly not least, I would like to thank my parents for their encouragement, and my wife for her understanding. Without their support, it would be impossible for me to finish my Ph.D. study, the most important step in my life.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The topic of this dissertation is a very specific area in information assurance: authentication techniques. This chapter reviews the information assurance model, prior work in the field of authentication, and summarizes the contributions of the dissertation.

1.1. Information Assurance Model

In the past decade, the understanding of information and system protection has evolved from Information Systems Security (INFOSEC) into Information Assurance (IA). According to the Information Assurance Glossary by the Committee on National Security Systems (CNSS) [23], Information Assurance has been defined as:

> Information operations (IO) that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. This includes providing for restoration of information systems by incorporating protection, detection and reaction capabilities.

This definition has been modeled by Maconachy et al. in [43], and a survey due to Loeb can be found at [41]. The model is shown in Fig 1, which is an expansion of the information security model due to McCumber proposed in 1991 [45].

The model presents three of the four dimensions of IA: Information States, Security Services, Security Countermeasures, and Time. The model illustrates that in any information system, information can be in one or more of the three states: stored, processing, or transmission. Security Countermeasures deal with the defense of information systems, which include measures in technology, operations, and people.

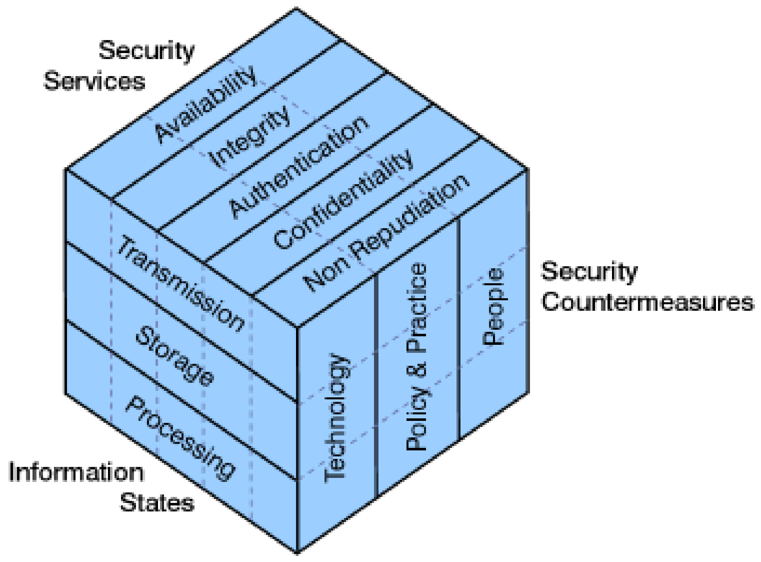

Figure 1. Information Assurance Model

Security Services are at the heart of information assurance. They include Availability, Integrity, Authentication, Confidentiality, and Non-repudiation. Availability ensures that authorized users can access data and information services timely and reliably. Integrity refers to the trustworthiness in information systems, such as data that can only be modified by authorized procedures, an information system in logical correctness, etc. Authentication is to establish the validity of procedures, or participants in an information system. It could be a means of verifying an individual's authorizations to receive specific categories of information. Confidentiality ensures that information can only be disclosed to authorized persons, processes, or devices. Non-Repudiation provides a mechanism to ensure that neither the sender of the data can deny what he has delivered, nor the receiver can repudiate his reception of such data.

1.2. Identification and Authentication

This dissertation investigates one of the five security services in the information assurance model: authentication. Authentication includes the validity of transmission, message, and originator. This dissertation focuses on methods to validate an originator in different settings, which is also called identification, or identity authentication.

According to [47], "identification is a statement of who the user is (globally known) whereas authentication is proof of identification." Thus, an authentication scheme would deal with how to construct such an "identification statement" and the method to prove such a statement. Authentication is fundamental to computing systems. Many security features are based on authentication. For instance, an access control mechanism might control access to system resources by a user's identity and security clearance level. To do so, the system must verify a user's identity before granting certain access rights.

Generally speaking, techniques for authentication fall into three categories [47]:

(i) Authentication by Knowledge ("Something You Know"): A user shows he knows something, such as password, to demonstrate his identity.

(ii) Authentication by Ownership ("Something You Have"): A user uses some physical token he has to show his identity, such as a credit card, or a driver's license.

(iii) Authentication by Characteristic ("Something You Are"): A user uses his biometric information to prove his identity, such as a fingerprint.

The combination of these three methods can provide strong authentication schemes. This dissertation mainly discusses authentication methods by "Something You Know", which is also the most widely used method in current distributed computing environments.

1.3. Authentication Techniques

The construction of most authentication schemes in information systems, just as their counterparts in daily life, relies on certain authorities trusted by all participants. These authorities issue credentials to legitimate users, and the presentation of a valid credential is the proof for a person's authenticity for certain behavior. For example, the driver's license system is essentially an authentication system, in which the government takes the responsibility to issue a valid credential, i.e., driver license, to a person who passes the driving test. Later, a person can show that he is authorized for driving by showing this credential when needed. Authentication in information systems follows the same principles.

However, additional difficulty exists in information systems. In an information system, a credential is nothing but a binary string, which can be easily duplicated, stolen, or modified. The challenge is how to construct a digital credential resistant to these attacks. Many authentication schemes have been proposed to construct such digital credentials with different features. This section reviews the main techniques in this area. Since most authentication techniques are based on public key cryptography, some basic concepts in public key cryptography will be introduced first.

1.3.1. Public Key Cryptography

The concept of public key cryptography was introduced by Diffie and Hellman in 1977 [26]. In a public key cryptosystem, a user Alice has a keypair $(PubKey, PrivKey)$, where $PubKey$ is publicly known, and $PrivKey$ is kept private by Alice. Suppose a remote party, Bob, wants to send a message m to Alice. For confidentiality of communication, Bob would encrypt this message as a ciphertext

$$c = E(PubKey, m),$$

where E is an encryption function. Function E takes $PubKey$ and m as input, and outputs a ciphertext c. Bob sends c to Alice. Alice recovers the plaintext as

$$D(PrivKey, c) = m,$$

where D is a decryption function. The security of a public key system relies on the property that a ciphertext can only be decrypted by Alice's private key, and only Alice knows her private key.

For some public key encryption schemes, a digital signature scheme may also be built based on this special keypair. Alice uses her private key to compute

$$sg = S(PrivKey, m),$$

where S is a sign function, and sg is called Alice's signature on message m. Alice sends (sg, m) to Bob, and Bob verifies whether

$$V(PubKey, sg) =? \; m,$$

where V is a verify function. If so, this shows sg is indeed Alice's signature on message m, since it is supposed that only Alice can compute sg based on her private key.

The invention of public key cryptosystems was a major breakthrough in cryptography. It opened the door for much of the research and applications of modern cryptography.

1.3.2. Digital Certificates

A digital certificate is a typical application of public key cryptography. In a digital certificate based authentication system, a trusted third party called a certificate authority (CA) issues digital certificates. The CA has a public key universally known by all participants. A user needs to apply for a certificate before he/she can authenticate himself/herself to others. The CA should ensure that only legitimate users can obtain certificates.

To apply for a certificate, Alice first generates her own public key and private key pair. Alice gives her public key to the CA while keeping her private key secret. The CA produces a string describing Alice's identity information. The string may also include other data related to certificate management such as an expiration date. The CA combines the identity string with Alice's public key, and signs the final string using its private key. The CA's signature together with Alice's public key and identity string is the digital certificate issued by the CA.

When Bob receives Alice's certificate, he first verifies the signature on the certificate to ensure that it is really issued by the CA. Then Bob uses Alice's public key to encrypt some random data. If Alice can decrypt this ciphertext, this proves Alice's authenticity. In many situations, Alice also needs to authenticate Bob, which leads to mutual authentication procedures.

The X.509 certificate standard [39] is an international standard based on this technique. X.509 has been adopted in the widely used Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocols which are used to protect most web applications [25]. X.509 is flexible, powerful, but also complex, and heavyweight. A typical certificate would be a separate object that is several times as large as the key that it is authenticating. For example, the current X.509 certificate authenticating the 128 byte (i.e., 1024 bit) key for www.amazon.com is 945 bytes long, so requires 738% overhead.

The deployment of X.509 incurs lots of effort at the infrastructure level. Therefore, it is mainly deployed on the Internet. In many situations, a simple and lightweight system is more desirable.

1.3.3. Identity-based Authentication

To simplify system design, the concept of identity-based public key cryptography was proposed by Shamir in 1984 [54]. In this scheme, the public key of a user is simply a string with his identity information. For example, Alice's email address could be her public key. However, Shamir only proposed a construction of an identity-based public key signature scheme, leaving the complete cryptographic system as an open problem. In 2001, Boneh and Franklin proposed a construction for a complete identity-based public key system [11].

In an identity based authentication system, a trusted third party, called the Key Generation Center (KGC), produces all the private keys for users. A very unusual characteristic for this type of system is that each user's private key is computed from the user's public key which is exactly the user's identity string, and this computation can only be carried out by the KGC based on its master key. No certificate exists in such a system. Therefore, the identity-based public key system is much simpler and easier to deploy than a certificate based system.

Identity-based authentication can implement a nice feature called "cryptographic workflow." Cryptographic workflow is a method to control the work process. For example, Alice sends Bob a message encrypted with a key string which is a specific date in the future. Bob can only get the decryption key from the KGC on the day specified by the encryption key. This technique has two implications: after Alice sends a message to Bob, she has no way to "cancel" her behavior because the decryption key can be obtained from the KGC based on the encryption key, the date. On the other hand, Bob can only open message when the specific day comes. This technique can carry out certain work management tasks which are hard to achieve by other cryptographic tools.

One major concern about identity-based public key systems is the key escrow problem, i.e., the KGC knows all private keys for the users. Therefore, the KGC can decrypt its users' ciphertexts. If the KGC is corrupted, or compromised, it could bring disaster to the whole system. This limitation makes the system more suitable for a constrained, small scope application area.

1.3.4. Self-certified Public Key

In 1991, Girault proposed an authentication scheme, called the self-certified public key scheme [34]. This scheme is an extension of the famous Schnorr identification scheme [52], with the additional support of authentication, making a separate certificate unnecessary.

In Girault's system, a user Alice and the KGC work together to produce a special construction called a witness combining Alice's public key and her identity. To some degree, the witness can be seen as a lightweight digital certificate [1]. The novel property for the self-certified public key system is that it avoids the key escrow problem in identity-based systems. Conceptually, it can be seen as a construction intermediate between a digital certificate system and an identity-based encryption system.

The advantage of the self-certified public key system is that it reduces a lot computation overhead and storage space while avoiding the key escrow problem exhibited by identity-based public key systems.

In 1997 Petersen and Horster explored the construction and applications of another construction of self-certified key scheme based on discrete logarithms modulo a prime number [49].

1.3.5. Certificateless Public Key

In 2003 Al-Riyami and Paterson proposed a new type of public key called a certificateless public key [1]. Their design is a modification of Boneh and Franklin's identity-based public key. It is similar to a self-certified public key while keeping the cryptographic workflow feature of identity-based public key cryptography. A user's public key is implicitly authenticated, so an additional certificate is unnecessary. It also avoids the key escrow problem in Boneh and Franklin's scheme.

The concept of a certificateless public key is very similar to a self-certified public key. Of course, their construction is quite different, and based on different security assumptions. Both of them are intermediate between certificate-based public keys and identity-based public keys. In [1] the authors provide a comparison between certificate-based public key and self-certified public key.

1.4. Anonymous Authentication

The wide spread applications on the Internet greatly change people's life patterns. While people enjoy the convenience of the Internet, one problem increasingly emerges as a big concern: privacy. People's personal data, such as address, birthday, social security number, hobbies, habits, etc, are scattered in numerous places on the Internet. Some data may be highly sensitive, while others could be private related. "Privacy protection" has become one of the most urgent problems to be solved these days.

Merriam-Webster defines privacy as "the quality or state of being apart from company or observation, freedom from unauthorized intrusion" [48]. Wikipedia defines privacy as "the ability of an individual or group to stop information about themselves from becoming known to people other than those they choose to give the information to" [60]. The authentication schemes introduced in the previous section are based on users' identities, or at most users' pseudonyms, without the consideration of privacy protection. Thus, a user's private data, scattered in different places, can be uniquely linked by this user's identity/pseudonym. Subsequently, a user's data could be collected, processed, and utilized without the awareness of its owner.

This raises the question: Can people be authenticated without revealing their real identities? This further leads to the concept of an anonymous authentication system in which a user can prove his/her authenticity without disclosing his/her identity. A user may also be able to show different appearances at different places to prevent link analysis. This technique, called Privacy-Through-Anonymity, has been widely explored in recently years in the cryptography area. Follows are the reviews for some main technologies in anonymous authentication
.

1.4.1. Group Signature

A group signature is a privacy-preserving signature scheme introduced by Chaum and Heyst in 1992 [22]. In such a scheme, a group member can sign a message on behalf of the group without revealing his identity. Only a specified open authority (who may or may not be the group manager) can open a signature and find its originator. Signatures signed by the same user cannot be identified as from the same source, i.e, "linked." Recently, the study of group signature schemes has attracted considerable attention, and many solutions have been proposed in the literature (e.g., [18, 17, 2, 3, 14, 9, 12]) [1].

[1]An extensive bibliography of group signature literature can be found at http://www.i2r.a-star.edu.sg/icsd/staff/guilin/bible/group-sign.htm

In a group signature scheme, a trusted party called the group manager issues group certificates to group members. The group certificate is a special construction jointly produced by the group manager and a group member through a protocol called the Join protocol. The Join protocol ensures that a valid group membership certificate can only be produced with the help of the group manager, and a group member knows a secret corresponding to this certificate. Later, only a group member can sign an anonymous signature based on his certificate if this secret is not exposed. The group signature scheme also provides a signature opening mechanism to reveal the signer.

Creating an anonymous authentication scheme from a group signature is simple: the group is simply the set of authorized users, and authentication is performed by a group member placing a group signature on a challenge (nonce) sent by the service requiring authentication. From the properties of group signatures, all the service or an attacker can learn is that the signature was made by a valid group member (i.e., an authorized user).

1.4.2. Traceable Signature

Unfortunately, anonymous authentication has a dangerous side effect: anonymous corrupted users. Since all users are anonymous, a mechanism is needed to identify corrupted users, effectively and fairly. In a group signature scheme, to reveal all behaviors by a malicious group member, the group manager has to open all the signatures in the pool. This is either inefficient (a centralized operation by the group manager), or unfair (unnecessarily identifying all innocent group members' signatures). To overcome this shortcoming, Kiayias et al. have recently proposed a variant of group signatures, called traceable signatures [40]. They define "traceability" as the ability to identify signatures signed by a specified group member (based on some per-user secret information called the "tracing trapdoor" kept by the group manager) without requiring the open authority to open them. Tracing can be done by "trace agents" distributively and efficiently. With a group signature, this cannot be done fairly since opening all signatures violates the privacy of innocent group members. Kiayias et al. also introduced the concept of "self-traceability," or "claiming." That is, a group member himself can stand out, claiming a signature signed by himself without compromising his other signatures and secrets.

1.4.3. Direct Anonymous Attestation

Another variant of group signatures has been adopted by the Trusted Computing Group [57], an industry group developing standards for "trusted computing platforms." A trusted computing platform is a computing device integrated with a cryptographic chip called the trusted platform module (TPM). The TPM is designed and manufactured in a specific way such that all parties can trust cryptographic computing results from this TPM. A trusted computing platform can implement many security related features based on the TPM, such as secure boot, sealed storage, and software integrity attestation.

However, deployment of TPMs introduces privacy concerns. Each TPM is loaded with its own unique public key pair called its "endorsement key," and the most straightforward use of a TPM would reveal this unique public key to remote parties during attestation operations. Thus, different servers could cooperate with each other to link the transactions made by the same TPM. To protect the privacy of a TPM owner, it is desirable to carry out anonymous authentication, i.e, a TPM can prove its authenticity to a remote server without disclosing its identifier. Such a mechanism has been implemented by the technique called Direct Anonymous Attestation (DAA) in [13]. DAA is basically a group signature scheme; however, since it is important for a user to have trust in such an anonymous attestation scheme, there is no open authority or capability for signatures to be opened.

In addition, DAA introduces the notion of "variable anonymity," which is conditionally linkable anonymous authentication: the same TPM will produce linkable signatures for a certain period of time. The period of time during which signatures can be linked can be determined by the parties involved and can vary from an infinitesimally short period (leading to completely unlinkable signatures) to an infinite period (leading to completely linkable signatures). Signatures made by the same user in different periods of time or to different servers cannot be linked. By setting the linkability period to a moderately short time period (a day to a week) a server can potentially detect if a key has been compromised and is being used by many different users, while still offering some amount of unlinkability.

1.5. Overview of this dissertation

This dissertation investigates authentication techniques for two typical applications in today's digital world: embedded systems, and pay web sites.

1.5.1. Application Background

Scenario 1: Embedded System. In this scenario, a person owns several devices which are capable of communication and interaction, but these devices use embedded processors whose computational capabilities are limited as compared to desktop computers. Examples of this scenario include entertainment devices or appliances owned by a consumer, multiple control and sensor systems in an automobile or airplane, and environmental controls in a building.

It would be desirable for these devices to use encrypted communication for confidentiality and integrity, and to be able to securely recognize other devices with the same owner. A pair of devices should be able to establish authenticated secure channels on their own, without needing to interact with the owner or any other party at the time, and the system is also dynamic, meaning that new devices can enter the system at any time.

Scenario 2: Pay Web Site. Today many web sites on the Internet provide pay services. To be able to access these services, a user may first be charged an annual fee to become a legitimate member with a login name and password. Later, this user can visit restricted pages by his/her login name and password.

However, users may be sensitive about their privacy, and not wish their behaviors to be tracked by web sites. Thus, an anonymous authentication scheme is desirable in this case. On the other hand, a web site certainly does not hope a bunch of users can share a single subscription. The website should be able to detect these possible malicious behaviors, and exclude corrupted users from future service.

1.5.2. Contributions of the Dissertation

The contributions of this dissertation include:

- An efficient public key cryptosystem for embedded systems. The proposed system is a complete solution for an embedded system, including authentication, authenticated key exchange, encryption and revocation protocols. The new construction is simple and efficient, and especially suitable for the devices with constrained computing capabilities and resources.

  Compared with digital certificate-based systems such as X.509, authentication is implicit in the key itself in the new scheme, so requires no overhead at all. Furthermore, the new scheme uses a short private exponent, allowing a device with a 1024 bit key to authenticate itself with roughly 15% of the number of modular multiplications required by an RSA signature, a common public key encryption algorithm used in X.509.

  The new scheme is more efficient than identity-based encryption [11]. More important than efficiency is the single weak point exhibited in IBE systems. That is, if the KGC is compromised then all user keys are immediately compromised. The new scheme can avoid this problem by adjusting the system parameters.

  Self-certified public keys [34, 49], certificate-less public keys [1], and group signature schemes [2] also provide similar capabilities to the proposed scheme. However, the basic model for these systems are quite different for target application, therefore many complications in these schemes are unnecessary in target settings.

  In addition, the proposed system supports a very compact revocation list and an efficient secure re-keying protocol. These features are not available in the current authentication scheme, but are particularly important for embedded systems.

- A group signature scheme with variable linkability which can be used as an anonymous authentication scheme. This construction follows a similar mechanism as in the first result to support anonymous authentication. The proposed group signature scheme supports two important features in anonymous authentication: signature claiming and variable anonymity, making it more practical in many settings. Another protocol is also proposed to carry out direct anonymous attestation for Trusted Computing Platforms. The new solution is a very simple and efficient construction, in which the Sign and Verify protocols are almost seven times more efficient than the current solution in [13].
- An efficient traceable signature which can be used as a more generic anonymous authentication scheme with the support of fair and efficient corrupted user tracing. This construction improves the state-of-the-art traceable signature scheme, and further supports variable linkability. As far as we know, this scheme is the first construction that supports a full revocation mechanism and variable anonymity. Table 1 presents a comparison of the features supported by different anonymous authentication techniques ($\surd$ for "supported", $\times$ for "not supported", and $-$ for "not available"). Clearly, the new scheme provides the most flexibility for group members as well as the group manager. Thus, it is the most suitable scheme for scenario 2: the pay web site.

The reason that these different authentication schemes can be put together into this dissertation lies in their common foundation. All these constructions are based on similar mathematical and cryptographic tools, sharing similar security assumptions. We call them flexible digital authentication schemes. Parts of these results have been accepted and will be published in [31, 32]

Scheme	Openability	Linkability	Traceability	Self-traceability
Pseudo-Anonymity	-	$\surd$	$\surd$	$\surd$
Group signature	$\surd$	$\times$	$\times$	$\times$
Traceable Signature	$\surd$	$\times$	$\surd$	$\surd$
Direct Anonymous Attestation	$\times$	variable	$\times$	$\surd$
The new scheme	$\surd$	variable	$\surd$	$\surd$

Table 1. Supporting Features of Anonymous Authentication Schemes

CHAPTER 2

MATHMATICAL AND CRYPTOGRAPHIC FOUNDATION

This chapter provides an introduction to some important concepts, theorems, and cryptographic primitives that will be used in the subsequent chapters. The aim of this chapter is to make this dissertation self-contained. Readers who are familiar with these topics may skip this chapter. References for these topics include the books by Shoup [56], Goldreich [35], and Menzes *et al* [46], and other research papers.

2.1. Abstract Algebra and Number Theory

This section describes the notions, theorems, and facts in abstract algebra and number theory that play core roles in the authentication schemes in this dissertation.

2.1.1. Groups and Cyclic groups

Definition 2.1.1. A group is a set G together with a binary operation * on G such that

- (Associative) for all $a, b, c \in G$, $a * (b * c) = (a * b) * c$,
- (Identity) there exists an $e \in G$ such that for all $a \in G$, $a * e = a = e * a$,
- (Inverse) for all $a \in G$, there exist $a^{-1} \in G$ such that $a * a^{-1} = e = a^{-1} * a$.

If a group satisfies the commutative property, i.e., for all $a, b \in G$, $a * b = b * a$, a group is called an *abelian group*. A group G is finite if $|G|$, the number of elements in G, is finite. $|G|$ is called the order of G.

Theorem 2.1.2. *Let G be an abelian group with binary operation *, then*

- *G contains only one identity element;*
- *every element of G has only one inverse.*

Definition 2.1.3 (Subgroup). A non-empty subset H of a group G is a subgroup of G if H is itself a group with respect to the operation to the operation of G.

Definition 2.1.4 (Cyclic group). A group G is cyclic if there is an element $g \in G$ such that for each element $a \in G$ there is an integer i with $a = g^i$. Such an element g is called a generator of G.

Theorem 2.1.5 (Lagrange's theorem). *If G is a finite group and H is a subgroup of G, then $|H|$ divides $|G|$.*

Definition 2.1.6. Let G be a group and $a \in G$. The order of a is the least positive integer t such that $a^t = e$, assuming that such an integer exists.

Theorem 2.1.7. *If G is a group and $a \in G$, then the set of all powers of a forms a cyclic subgroup of G, called the subgroup generated by a, and denoted by $\langle a \rangle$.*

2.1.2. The Group Z_n^*

For a positive integer n, and $a, b \in Z$, if $a-b$ can divide n, a is said congruent to b modulo n, and written as $a \equiv b \pmod{n}$. Obviously, $a \equiv b \pmod{n}$ if and only if $a = b + cn$ for an integer c. The modulo operation maps all integers into the set of $(0, n-1)$, which leads to the technique known as modular arithmetic. Modular arithmetic exhibits the following properties:

- $(a \pmod{n} + b \pmod{n}) \pmod{n} = (a + b) \pmod{n}$
- $(a \pmod{n} - b \pmod{n}) \pmod{n} = (a - b) \pmod{n}$
- $(a \pmod{n} \times b \pmod{n}) \pmod{n} = (a \times b) \pmod{n}$

Definition 2.1.8 (Multiplicative Group Z_n^*). The set of all positive integers that are less than n and relatively prime to n, together with the operation of multiplication modulo n, forms a multiplicative group Z_n^*.

Definition 2.1.9 (Euler's Totion Function). $\phi(n) = |Z_n^*|$.

- If p is a prime, then $\phi(p) = p - 1$.
- The Euler totion function is multiplicative. That is, if gcd(m,n) = 1, then $\phi(mn) = \phi(m)\phi(n)$.
- If $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ is the prime factorization of n, then

$$\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k}).$$

2.1.3. Fermat's Theorem and Euler's Theorem

Theorem 2.1.10 (Fermat's Theorem). *If p is prime and a is a positive integer not divisible by p, then*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Theorem 2.1.11 (Euler's Theorem). *For every a and n that are relatively prime,*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

2.1.4. Euclid's Algorithm

Theorem 2.1.12 (Euclid's theorem). *For any non-negative integer a and any positive integer b,*

$$GCD(a, b) = GCD(b,\ a \pmod{b}). \tag{1}$$

Algorithm 2.1.1 (Basic Euclidean Algorithm). For any non-negative integer a, b, one can repeat using *equation* 1 to find the greatest common divisor d of a, b such that $d = GCD(a, b)$ in $O(len(a) + len(b))$ arithmetic operations.

Algorithm 2.1.2 (Extended Euclidean Algorithm). For any non-negative integer a, b, let $d = GCD(a, b)$. There exists integers s, t such that $as + bt = d$. The Extended Euclidean Algorithm allows us to efficiently compute s and t in $O(len(a)+len(b))$ arithmetic operations.

2.2. Number-Theoretic Problems

The security of many cryptographic systems relies on the assumption that certain computational problems are intractable. The *integer factorization problem* and *discrete logarithm problem* are two fundamental problems for many cryptographic systems. The intractability of these two problems is still an assumption, since no one has been able to prove their hardness. In fact, a proof that either of these problems cannot be solved in polynomial time would give as a consequence that $P \neq NP$, solving what is arguably the most famous and important open problem in computer science. Nonetheless, with many years of experience examining these problems, they are widely accepted as intractable problems in cryptographic literature. The security of the schemes in this dissertation also relies on the hardness of these two problems as well as additional problems that will be discussed in the next chapter.

2.2.1. The Integer Factorization Problem

Definition 2.2.1 (Integer Factorization Problem). Given a positive integer n, find its prime factorization; that is, write $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.

The current available algorithms for factoring an integer n include [1]:

- Pollard's rho algorithm with running time $O(\sqrt{n})$.
- Elliptic curve method: if n has a small prime factor p, this method finds p in $O(\exp((1 + o(1))\sqrt{2 \ln p \ln \ln p}))$.
- Quadratic sieve algorithm with running time $O(\exp((1 + o(1))\sqrt{\ln n \ \ln \ln n}))$.
- Number field sieve algorithm with running time $O(\exp((1.92+o(1))(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}))$.

[1]Note that following the convention of computational number theory, n is used to represent the number itself rather than the length of the number, as is standard in other areas of algorithms. This means that an $O(n)$ time algorithm is actually exponential time.

All the current integer factorization algorithms are super-polynomial, and it is widely accepted that this problem is intractable. Therefore the integer factorization assumption is described as follows.

Assumption 2.2.2 (Integer Factorization Assumption). There exists no probabilistic polynomial-time algorithm that can find the factors of an arbitrary integer with non-negligible probability.

2.2.2. The Discrete Logarithm Problem

Definition 2.2.3 (Discrete Logarithm). Let G be a finite cyclic group and $g \in G$ be a generator of G. The discrete logarithm of an element $a \in G$, is the unique integer $x, 0 \leq x < |G|$, such that $a = g^x$.

Definition 2.2.4 (Discrete Logarithm Problem). Given a finite cyclic group G, a generator g of G, and an element a, find the integer $x, 0 \leq x < |G|$, such that $a = g^x$.

The current available algorithms for discrete logarithm problems include:

- Generic algorithms that work in arbitrary groups that include Pollard's rho algorithm, and the Baby-step Giant-step algorithm. Both run in time $O(\sqrt{n}\log(n))$
- Pohling-Hellman algorithm for a special group whose order has only small prime factors. It's running time is $O(\sum_{i=1}^{r} e_i(\lg n + \sqrt{p_i}))$.
- Index calculus algorithm for groups Z_p^* and $Z_{2^m}^*$ with running time upper-bound by $O(\exp((c + o(1))\sqrt{\ln q \ln\ln\ q}))$.
- The number field sieve method for group Z_p^* runs with time $O(\exp((1.92+o(1))(\ln p)^{\frac{1}{3}}(\ln\ln p)^{\frac{2}{3}}))$.

It is widely accepted that if fro a suitable group, the discrete logarithm problem is intractable. This is expressed as the following assumption.

Assumption 2.2.5 (Discrete Logarithm Assumption). There exists no probabilistic polynomial-time algorithm that can find the discrete logarithm of an element in a group G that supports the discrete logarithm assumption with non-negligible probability.

Many groups support the discrete logarithm assumption. For example, the discrete logarithm problem is assumed intractable in a group Z_p^* where p has a large factor.

2.3. Hash Function

Hash functions are used in many cryptographic schemes such as digital signatures and authentication.

Definition 2.3.1 (Hash Function). A hash function is a function mapping binary strings of arbitrary finite length to binary strings of fixed length l:

$$\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l.$$

For cryptographic purposes a hash function must also be hard to invert, i.e., it must have at least one of the following properties (in decreasing order of desirability):

- *Strong collision resistant*: It is hard to find a pair (x, x') with $x \neq x'$ such that $\mathcal{H}(x) = \mathcal{H}(x')$.
- *Weak collision resistant*: For a given x, it is hard to find an $x' \neq x$ such that $\mathcal{H}(x) = \mathcal{H}(x')$.
- *One-way*: For a given c, it is hard to find an x such that $c = \mathcal{H}(x)$.

2.4. The Public Key Cryptosystem

Chapter 1 gave a high level introduction to the mathematics behind much of public key cryptography. This section introduces three famous schemes in public key cryptography: the Diffie-Hellman key exchange scheme, the RSA public key scheme, and the ElGamal public key scheme.

2.4.1. The Diffie-Hellman Key Exchange Scheme

In 1976, Diffie and Hellman introduced the concepts of public key cryptosystems in their pioneering paper "New Direction In Cryptography" [26], and proposed a key exchange scheme based on this novel concept. The Diffie-Hellman key exchange protocol allows Alice and Bob to generate a session key for later communication in such a way that other parities can not calculate this session key by observing the data exchange between Alice and Bob. The protocol works as follows:

- Alice and Bob agree on a large prime number p and a primitive root g of p, which is a generator of the group Z_p^*. For security reasons, $p-1$ should have at least one large prime factor.
- Alice randomly picks x in $(1, p-1)$, denoted by $x \in_R (1, p-1)$, and calculates $A = g^x \pmod{p}$; Bob randomly picks $y \in_R (1, p-1)$ and calculate $B = g^y \pmod{p}$. Alice and Bob exchange A, B.
- Alice calculates

  $$B^x = (g^y)^x = g^{xy} \pmod{p},$$

  and Bob calculates

  $$A^y = (g^x)^y = g^{xy} \pmod{p}.$$

Since Alice knows her secret x, and Bob knows his secret y, Alice and Bob can complete the above computation. It has become a widely accepted assumption that computing $g^{xy} \pmod{p}$ given just A and B, known as the Computational Diffie-Hellman Problem, is infeasible. A formal definition of the Computational Diffie-Hellman Problem will be given in the next chapter. Thus, Bob and Alice share a common secret from which a session key can be generated.

The Diffie-Hellman key exchange protocol elegantly solves the key exchange problem. The pioneering work of Diffie and Hellman opened the door to research and applications of public key cryptography. However, the protocol does not provide authentication, and suffers a "man-in-the-middle" attack, in which a third party sitting in the middle of Alice and Bob can pretend he/she is Alice to Bob, and Bob to Alice, without the awareness of Alice and

Bob. The Diffie-Hellman key exchange protocol should be combined with other cryptographic techniques to perform authenticated key exchange.

2.4.2. The RSA Scheme

In 1977, Rivest, Shamir and Addleman proposed the first, still widely used public key cryptosystem [51]. The following gives a high level description of this famous construction. Note that this is an abstract and high-level description - in practice, the use of RSA is quite different.

Alice picks two large random prime numbers p, q. She computes $n = pq$, and Euler's totient function $\phi(n) = (p-1)(q-1)$. She further picks an $e < n$ that is relatively prime to $\phi(n)$, and computes d, the inverse of e modulo $\phi(n)$. While the original RSA paper suggested a random e, no security weaknesses have been discovered with using small fixed e, as long as p and q satisfy some additional properties. So in practice e is almost always either 3 or 65537, since this leads to much faster encryption operations. Then (d, n) is the private key of Alice, and (e, n) is the public key. $(d, p, q, \phi(n))$ are kept secret by Alice. If another user Bob wants to send a message $m < n$ to Alice, he computes the ciphertext

$$c = m^e \pmod{n},$$

and sends it to Alice. Alice decrypts the ciphertext as

$$m' = c^d \pmod{n}.$$

If Alice and Bob follow the protocol, since Z_n^* is a group with order $\phi(n)$, and $ed = k\phi(n)+1$ for some integer k, then

$$m' = c^d = (m^e)^d = m^{ed} = m \pmod{n}.$$

A digital signature scheme can also be implemented in the RSA scheme. Alice uses her private key d to compute

$$sg = m^d \pmod{n},$$

where sg is Alice's signature on message m. Bob can verify

$$sg^e =?\ m \pmod{n}.$$

If so, this shows sg is indeed Alice's signature on message m, since it is supposed that only Alice can compute sg based on her private key d.

2.4.3. The ElGamal Scheme

ElGamal proposed a public key cryptosystem based on discrete logarithms modulo a large prime number in [27]. In this system, Alice has her private key $x \in_R (1, p-1)$, and public key $y = g^x \pmod{p}$. p is a large prime number such that $p-1$ has at least one large prime factor. g is a generator of group Z_p^*. Bob wants to send a message m $(0 < m < p-1)$ to Alice. He randomly chooses $k \in_R (1, p-1)$, and computes the key

$$K = y^k = g^{xk} \pmod{p},$$

and the ciphertext is the pair (c_1, c_2), where

$$c_1 = g^k \pmod{p},\ \ c_2 = Km \pmod{p}.$$

To decrypt the message (c_1, c_2), Alice calculates

$$K = c_1^x = g^{xk} \pmod{p},\ \ m = c_2 K^{-1} \pmod{p}.$$

Since only Alice is supposed to know her private key, under appropriate assumptions only she can decrypt the message to her. A digital signature scheme can also be implemented using ElGamal's techniques. Readers should refer to the original paper for details.

2.5. The Random Oracle Paradigm

The random oracle model is a common technique in cryptography, and was introduced by Goldreich et al. [36, 37] and Fiat-Shamir [28], and explicitly formalized by Bellare and Rogaway [6]. It has been widely used to prove security of cryptographic protocols, and many famous cryptographic schemes have been proved secure in the random oracle model. For example, the RSA scheme with the Optimal Asymmetric Encryption Padding (OAEP), which is one way the RSA scheme is used in practice, has been proved secure in the random oracle model [7, 55].

A random oracle is a mathematical abstraction, which answers a query x as follows:

- If it answered the query x before, it responds with the same value it gave the last time.
- If it has not answered the query x before, it generates a random response which has uniform probability of being chosen from anywhere in the oracle's output domain. The random oracle also records the answer for this x in case the same query is asked later.

In the random oracle model, a public random oracle can be accessed by all parities, either good or bad. A protocol is firstly proved correct in this model. Then the random oracle is replaced by a hash function, as is common practice, but note there are some potential problems with this substitution [19].

The schemes in this dissertation will be proved secure in the random oracle model.

2.6. Zero-knowledge Proof of Knowledge

Zero-knowledge proofs were introduced by Goldwasser, Micali and Rackoff in [38]. It forms a central class of modern cryptographic protocols. Through a zero-knowledge protocol, a prover can convince a verifier of the validity of a statement in an interactive mode, while the verifier can not gain any information from the interaction with the prover except the truth of statement itself. This section reviews some important definitions about zero-knowledge proofs.

2.6.1. Interactive Proof System

Traditionally, a proof of a statement is a sequence of statements. Anyone with knowledge in this area should be able to interpret these statements and judge the validity of the statement to be proved. In contrast, an interactive proof of a statement is an interactive protocol between a prover and a verifier. The prover wants to convince the verifier of the validity of a statement. For example, the statement can be of the form "Formula ϕ is satisfiable," or "Graph G is three-colorable." Two requirements are necessary for a valid interactive proof system: completeness and soundness. The completeness property deals with the ability of an honest prover, i.e., if the statement is true, the prover should be able to convince the verifier with high probability. The soundness property deals with the inablility of a cheater, i.e., if the statement is false, no prover should be able to convince a verifier with significant probability. Let's recall the definition of interactive proof system that appears as definition 4.2.4 in [35]. In the following, $\langle A, B\rangle(x)$ denotes the random variable representing the output of B when interacting with A on common input x, when the random input to A and B is uniformly and independently chosen.

Definition 2.6.1. (Interactive proof system). A pair of interactive Turing machines (P,V) constitute an interactive proof system for language L if machine V is polynomial-time and the following two conditions hold:

- Completeness: For every $x \in L$,

$$Pr[\langle P, V\rangle(x) = 1] \geq \frac{2}{3}.$$

- Soundness: For all $x \notin L$ and all interactive Turing machine B,

$$Pr[\langle B, V\rangle(x) = 1] \leq \frac{1}{3}.$$

An interactive proof system can be of interest only if the verifier is probabilistic polynomial-time, which is reflected in the above definition. Otherwise, a verifier with powerful computation capability may be able to make the decision alone without interacting with a prover, i.e, it can determine whether $x \in L$ alone. However, in practice, the prover and verifier in general should both be probabilistic polynomial-time. Thus the question arises of how a prover can have more capability than a verifier to complete an interactive proof. In reality, the prover's power comes from the fact that the prover also takes some auxiliary input. That is, a polynomial-time machine P is equipped with some knowledge which is not known by a verifier. Definition 2.4.2 from [42] reflects this situation.

Definition 2.6.2. (Interactive argument). A pair of interactive probabilistic polynomial-time Turing machines (P,V) constitute an interactive proof system for language L if:

- Completeness: If $x \in L$, then there exists a witness w such that

$$Pr[\langle P(w), V\rangle(x) = 1] = 1,$$

  where the prover has additional input w.
- s-Soundness: For all $x \notin L$ and all interactive Turing machine B, $\Pr[\langle B, V\rangle(x) = 1] \leq s(|x|)$.

2.6.2. Zero-knowledge Proofs

In the design of cryptographic protocols, it would be desirable for an interactive proof system being "zero-knowledge." Intuitively, a proof-system is zero-knowledge if a third party alone can produce a transcript indistinguishable from the transcript produced between the prover and the verifier. Precisely speaking [35]: Let (P, V) be an interactive proof system for some language L. If for every probabilistic polynomial-time interactive machine V^* there exists a probabilistic polynomial-time machine M^* such that for every $x \in L$ random variables

$\langle P, V^* \rangle(x)$ and $M^*(x)$ are identically distributed, (P, V) is called a perfect zero-knowledge proof system. Machine M^* is called a simulator for the interaction of V^* with P.

This dissertation will use a weaker notion of zero-knowledge, called honest-verifier zero-knowledge, to construct authentication protocol. This notion requires simulatability of the view of only the prescribed (or honest) verifier, rather than simulatability of the view of any possible verifier. Let's first review the concept of computational indistinguishability. For two probability ensembles $\{R_x\}_{x \in L}$and $\{S_x\}_{x \in L}$, if for every probabilistic polynomial-time algorithm D, for every polynomial $p(\cdot)$, and for all sufficiently long $x \in L$, it holds that

$$|Pr[D(x, R_x) = 1] - Pr[D(x, S_x) = 1| < \frac{1}{p(|x|)},$$

these two probability ensembles are called computational indistinguishable [35]. The following is the definition of zero-knowledge proof with respect to an honest verifier (Definition 4.3.7 in [35]).

Definition 2.6.3. (Zero-knowledge with respect to an Honest Verifier). Let (P, V) be a pair of polynomial-time machines and L be a language. $view_V^P(x)$ is a random variable describing the content of the random tape of V^* and the messages V^* receives from P during a joint computation on common input x. It is said that (P, V) is honest-verifier zero-knowledge if there exists a probabilistic polynomial-time algorithm M such that the ensembles $\{view_V^P(x)\}_{x \in L}$ and $\{M(x)\}_{x \in L}$ are computationally indistinguishable.

2.6.3. Proofs of Knowledge

The previous section has reviewed the concept of interactive proof systems, and the property of zero-knowledge. It is further interesting if a proof system can demonstrate that a certain quantity w that satisfies some polynomial-time computable relation R is known to the prover. This leads to a proof of knowledge. It should be noted that a knowledge proof is not necessarily a zero-knowledge proof. Let's review the definition for proof of knowledge in [42].

Definition 2.6.4. (Proof of knowledge). Let $R(\cdot, \cdot)$ be a polynomially computable relation. Let $u(x)$ be such that $|w| \leq u(x)$ for all w such that $R(x, w)$ holds, and assume that $u(x) = poly(|x|)$ is efficiently computable. A verifier V is a knowledge verifier with respect to R if:

- Non-triviality: There exists a prover P such that for all x, w, if $R(x, w) = 1$, then

$$Pr[\langle P(w), V\rangle(x) = 1] = 1$$

- Extraction with knowledge error $2^{-u(x)}$: There exists an extractor algorithm K and a constant c such that for all x, for all adversaries A, if

$$p(x) = Pr[\langle A, V(x)\rangle = 1] > 2^{-u(x)}$$

  then, on input x and with access to prover, K computes a value w such that $R(x, w)$ holds, within an expected number of steps bounded by $\frac{(|x|+u(x))^c}{p(x)-2^{-u(x)}}$.

2.7. Authentication Techniques

This dissertation will adopt two authentication techniques to construct authentication protocols in different settings. The first technique is based on public key cryptosystems, while the second technique is based on more powerful zero-knowledge proof systems. Generally speaking, authentication based on public key techniques is more simple and efficient than one based on a zero-knowledge proof system. However, a zero-knowledge proof system can be more powerful for implementing a complex authentication system, especially for an anonymous authentication system.

2.7.1. Authentication based on Public Key Technique

Two methods are available for implementing authentication based on public key techniques: encryption and digital signature. To use an encryption scheme, Bob challenges Alice using a ciphertext which is a random message encrypted using Alice's public key. Only Alice can decrypt the ciphertext and correctly answer Bob's challenge. To use a digital signature scheme, Alice signs some one-time message (e.g., timestamps or a nonce) using her private key and sends the result to Bob. Bob verifies that the message does originate from Alice using Alice's public key.

The procedure below is a specific procedure for authentication by public key encryption.

- Bob picks a random r. He computes a witness $w = \mathcal{H}(r)$ and a challenge $c = E(PubKey_a, r)$, and sends (w, c) to Alice.
- Alice uses her private key to compute $r' = D(PriKey_a, c)$. Alice checks $\mathcal{H}(r') =? w$. If not, Alice aborts the protocol. Otherwise, Alice sends r' to Bob.

$\mathcal{H}$ is a one-way hash function. w is called a witness, which is used to prevent chosen message attacks from malicious challengers. In a chosen message attack, an attacker could forge some special message m, and trick Alice to compute m^d. Potentially, this might reveal certain information for Alice's private key. With the help of a witness, Alice can detect such attacks and abort the protocol. On the other hand, if Bob is honest, the response from Alice is already known by himself. Therefore, nothing about Alice's private key will be revealed.

2.7.2. Authentication based on a Zero-knowledge Proof of Knowledge

In constructing authentication scheme based on a zero-knowledge proof system, Alice sends Bob a commitment to her secret. Then Alice proves to Bob that she knows the secret in the commitment without revealing it, and that the secret satisfies certain properties. This methodology will be demonstrated in the next chapter with a real example. In this dissertation, this technique is extensively used to implement anonymous authentication in different settings.

CHAPTER 3

PRELIMINARIES

This chapter reviews the definitions, assumptions and building blocks which will be used throughout this dissertation.

3.1. Definitions

Definition 3.1.1 (Special RSA Modulus[15]). An RSA modulus $n = pq$ is called special if $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' also are prime numbers. p, q are called safe primes. A special RSA modulus is also called a safe RSA modulus in the literature (for example, [2]).

Definition 3.1.2 (Quadratic Residue Group QR_n). Let Z_n^* be the multiplicative group modulo n, which contains all positive integers less than n and relatively prime to n. An element $x \in Z_n^*$ is called a *quadratic residue* if there exists an $a \in Z_n^*$ such that $a^2 \equiv x \pmod{n}$. The set of all quadratic residues of Z_n^* forms a cyclic subgroup of Z_n^*, denoted by QR_n. If n is the product of two distinct primes, then $|QR_n| = \frac{1}{4}|Z_n^*|$.

Property 3.1.1. If n is a special RSA modulus, with p, q, p', and q' as in Definition 3.1.1 above, then $|QR_n| = p'q'$ and $(p'-1)(q'-1)$ elements of QR_n are generators of QR_n.

Proof: Consider the group Z_p^* and corresponding subgroup QR_p. Since p is prime, exactly half of the elements of Z_p^* are in QR_p, so $|QR_p| = \frac{1}{2}(p-1) = p'$. Since p' is prime, any element of QR_p generates a subgroup of size 1 or of size p', and since only the identity element generates a subgroup of size 1, the remaining $p'-1$ elements of QR_p have order p'. Similarly, $q'-1$ elements of QR_q have order q'. By the Chinese Remainder Theorem, if element x_p has order m_p in Z_p^* and element x_q has order m_q in Z_q^*, then the unique element

$x \in Z_n^*$ with $x_p = x \bmod p$ and $x_q = x \bmod q$ has order $\mathrm{LCM}(m_p, m_q)$ in Z_n^*. Furthermore, $x \in QR_n$ if and only if $x_p \in QR_p$ and $x_q \in QR_q$. Therefore, $(p'-1)(q'-1)$ elements of QR_n have order $\mathrm{LCM}(p', q') = p'q' = |QR_n|$ in QR_n, and so are generators of QR_n. □

The following two properties follow from basic properties of cyclic groups and quadratic residues.

Property 3.1.2. If g is a generator of QR_n, then $g^a \bmod n$ is a generator of QR_n if and only if $\mathrm{GCD}(a, |QR_n|) = 1$.

Property 3.1.3. If $x \in_R Z_n^*$ is uniformly distributed over Z_n^*, then $x^2 \bmod n$ is uniformly distributed over QR_n.

The concept of negligible function (Definition 1.3.5 in [35]) is defined as follows.

Definition 3.1.3 (Negligible Function). A function $v(k)$ is negligible, if for any polynomial $p(\cdot)$, there exist an N such that all $k > N$,

$$v(k) < \frac{1}{p(k)}.$$

3.2. Number-Theoretic Assumptions

The security of the techniques in this dissertation relies on the following assumptions, which are widely accepted in the cryptography literature (see, for example, [5, 29, 8]).

Assumption 3.2.1 (Strong RSA Assumption). Let n be an RSA modulus. The *Flexible RSA Problem* [1]is the problem of taking a random element $u \in Z_n^*$ and finding a pair (v, e) such that $e > 1$ and $v^e = u \pmod n$. The *Strong RSA Assumption* says that no probabilistic polynomial time algorithm can solve the flexible RSA problem with non-negligible probability.

[1]The flexible RSA problem is also called the strong RSA problem in some literature (e.g. [2]).

The strong RSA assumption strengthens the widely accepted RSA assumption that finding the e^{th}-roots modulo n for an element in Z_n, where e is the fixed public exponent. More discussion can be found in the report by Rivest and Kaliski [50].

The following lemmas will be used intensively for the security proofs in the rest of this dissertation. The first lemma is due to Shamir [53].

Lemma 3.2.2. *Let n be an integer. For given values $u, v \in Z_n^*$ and $x, y \in Z_n$ such that $GCD(x, y) = 1$ and $v^x \equiv u^y \pmod{n}$, there is an efficient way to compute the value z such that $z^x \equiv u \pmod{n}$.*

Proof: Since $GCD(x, y) = 1$, one can use the Extended GCD algorithm to find a and b such that $ay + bx = 1$, and let $z = v^a u^b$. Thus

$$z^x \equiv v^{ax} u^{bx} \equiv u^{ay+bx} \equiv u \pmod{n}.$$

□

Shamir's lemma can be extended as follows.

Lemma 3.2.3. *Let n be an integer. Given values $u, v \in Z_n^*$ and $x, y \in Z$ such that $GCD(x, y) = r$, and $v^x \equiv u^y \pmod{n}$, there is an efficient way to compute a value z such that $z^k \equiv u \pmod{n}$, where $k = x/r$.*

Proof: Since $GCD(x, y) = r$, using the extended Euclidean GCD algorithm, one can obtain values α and β such that $\alpha x/r + \beta y/r = 1$. Thus

$$u \equiv u^{\alpha x/r + \beta y/r} \equiv u^{\alpha x/r} u^{y\beta/r} \equiv u^{\alpha x/r} v^{\beta x/r} \equiv (u^\alpha v^\beta)^{x/r} \pmod{n}.$$

Therefore, setting $k = x/r$ and $z = u^\alpha v^\beta$, one can have $z^k \equiv u \pmod{n}$. □

The following lemma is based on the strong RSA assumption.

Lemma 3.2.4. *Under the strong RSA assumption, if there exists a probabilistic polynomial-time algorithm that takes an RSA modulus n and a value u and succeeds with non-negligible probability in finding values v, x, and y, such that $v^x \equiv u^y \pmod{n}$, then x divides y.*

Proof: By contradiction. Assume that there exists a probabilistic polynomial-time algorithm that takes an RSA modulus n and a value u and succeeds with non-negligible probability in finding values v, x, and y, such that $v^x \equiv u^y \pmod{n}$, but for which x does *not* divide y. Let $r = GCD(x, y)$. Since x does not divide y one have $r < x$, and so $x/r > 1$. By Lemma 3.2.3 one can find a z such that $z^k = u \pmod{n}$, with $k = x/r > 1$, which is a solution to the flexible RSA problem. However, this contradicts the strong RSA assumption, which says that no such algorithm can exist. □

Some additional security assumptions are introduced as follows.

Assumption 3.2.5 (Computational Diffie-Hellman Assumption for QR_n). Let n be a special RSA modulus, and let g be a generator of QR_n. Then given random g^x and g^y, it is hard to compute $g^{xy} \pmod{n}$.

Assumption 3.2.6 (Decisional Diffie-Hellman Assumption for QR_n). Let n be a special RSA modulus, and let g be a generator of QR_n. For two distributions (g, g^x, g^y, g^{xy}), (g, g^x, g^y, g^z), $x, y, z \in_R Z_n$, there is no probabilistic polynomial-time algorithm that distinguishes them with non-negligible probability.

Kiayias et al. have investigated the Decisional Diffie-Hellman Assumption over a subset of QR_n in [40], i.e., x, y, z are randomly chosen from some subsets, truncation of QR_n. They showed that the Decisional Diffie-Hellman Assumption is still attainable over subsets of QR_n with the size down to at least $|QR_n|^{1/4}$.

The following lemma is a corollary of the decisional Diffie-Hellman assumption.

Lemma 3.2.7. *Let n be a special RSA modulus, g be a generator of QR_n, and three elements $a, b, c \in QR_n$. There is no probabilistic polynomial-time algorithm that can decide with non-negligible probability whether there exists an integer x such that $a = g^x, c = b^x$.*

Proof: Suppose there exists a probabilistic polynomial-time algorithm $\mathcal{A}(g, a, b, c)$ for $a, b, c \in QR_n$ and g being a generator of QR_n, which can decide there exist an integer x such that $a = g^x, c = b^x$. Consider two distribution distributions (g, g^x, g^y, g^{xy}), (g, g^x, g^y, g^z), $x, y, z \in_R Z_n$. One can call $\mathcal{A}(g, g^x, g^y, g^{xy})$, and $\mathcal{A}(g, g^x, g^y, g^z)$ to recognize with non-negligible probability which distribution is (g, g^x, g^y, g^{xy}) since this distribution can be recognized by the algorithm $\mathcal{A}$ due to $g^{xy} = (g^y)^x$, and $g^x = (g)^x$. However, this is infeasible under the decisional Diffie-Hellman assumption. Therefore, no such algorithm $\mathcal{A}$ exits. $\square$

3.3. Building Blocks

The main building blocks in Chapter 5 and 6 of this dissertation are *statistical honest-verifier zero knowledge proofs of knowledge* related to discrete logarithms over QR_n [29, 30, 16] which will be reviewed in this section. They include protocols for things such as knowledge of a discrete logarithm, knowledge of equality of two discrete logarithms, and knowledge of a discrete logarithm that lies in an interval, etc. These protocols have been proved secure under the strong RSA assumption, and proofs can be found in the cited references.

3.3.1. Knowledge of a Discrete Logarithm

Fujisaki and Okamota proposed a zero-knowledge proof of knowledge of a discrete logarithm [29] . Camenisch and Michels proposed a slight modification and provided a security proof in [16].

Protocol 3.3.1. Let n be a special RSA modulus, QR_n be the quadratic residue group modulo n, and g be a generator of QR_n. l_g is the bit length of $|QR_n|$. α and l_c are security parameters that are both greater than 1. A prover Alice knows x, the discrete logarithm of T (so $g^x = T$). Alice demonstrates her knowledge of x as follows.

(i) Alice picks a random $t \in \pm\{0,1\}^{\alpha(l_g+l_c)}$ and computes $d = g^t \pmod{n}$. Alice sends (T, d) to verifier Bob.

(ii) Bob picks a random $c \in \{0,1\}^{l_c}$ and sends it to Alice.

(iii) Alice computes

$$w = t - cx,$$

and $w \in \pm\{0,1\}^{\alpha(l_g+l_c)+1}$. Alice sends w to Bob.

(iv) Bob checks $w \in \pm\{0,1\}^{\alpha(l_g+l_c)+1}$ and

$$g^w T^c =?\ d \pmod{n}.$$

If the equation holds, Alice proves knowledge of the discrete logarithm of T.

Remark 3.3.1. The parameter $\alpha > 1$ is used since the size of the group QR_n is unknown, and determines the statistical closeness of the actual distribution to the ideal one. In other words, α determines the statistical zero-knowledge property of this protocol. This will be illustrated more clearly in the proof for the protocol of knowledge of a discrete logarithm in an interval (Protocol 3.3.3).

Remark 3.3.2. Using the Fiat-Shamir heuristic [28], the protocol can be turned into a non-interactive "signature of knowledge," which is secure in the random oracle model [6].

3.3.2. Knowledge of the Equality of Discrete Logarithms with Different Bases

The protocol given below was first introduced in [21]. It was adapted to work in a group with unknown order in [16]. In this protocol, Alice proves she knows discrete logarithms of two elements $T_1, T_2 \in QR_n$ with respect to generators g, h, respectively. That is, $g^x \equiv T_1 \pmod{n}$, $h^x \equiv T_2 \pmod{n}$. The protocol works as follows.

Protocol 3.3.2. Let n be a special RSA modulus, QR_n be the quadratic residue group modulo n, and g be a generator of QR_n. l_g is the bit length of $|QR_n|$. α and l_c are security parameters that are both greater than 1.

- Alice picks a random $t \in \pm\{0,1\}^{\alpha(l_g+l_c)}$ and computes

$$d_1 = g^t \pmod{n},\ d_2 = h^t \pmod{n}.$$

  Alice sends (T_1, T_2, d_1, d_2) to verifier Bob.
- Bob picks a random $c \in \{0,1\}^{l_c}$ and sends it to Alice.

- Alice computes

$$w = t - cx,$$

and $w \in \pm\{0,1\}^{\alpha(l_g+l_c)+1}$. Alice sends w to Bob.

- Bob checks $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$ and

$$g^w T_1^c =?\ d_1 \pmod{n},\ \ h^w T_2^c =?\ d_2 \pmod{n}$$

If the equation holds, Alice has proved knowledge of equality of two discrete logarithms of T_1, T_2 with base g, h.

3.3.3. Knowledge of a Discrete Logarithm in an Interval

The protocol was first proposed for a group with known order by Chan et al. in [20]. Fujisaki and Okamota extended the protocol to a group with unknown order in [30]. Camenisch and Michels proposed a slight modification in [17].

Protocol 3.3.3. Let n be a special RSA modulus, QR_n be the quadratic residue group modulo n, and g be a generator of QR_n. α, l, l_c are security parameters that are all greater than 1. X is a constant number. A prover Alice knows x, the discrete logarithm of T, and $x \in [X - 2^l, X + 2^l]$. Alice demonstrates her knowledge of $x \in [X - 2^l, X + 2^l]$ as follows.

(i) Alice picks a random $t \in \pm\{0,1\}^{\alpha(l+l_c)}$ and computes $d = g^t \pmod{n}$. Alice sends (T, d) to a verifier Bob.

(ii) Bob picks a random $c \in \{0,1\}^{l_c}$ and sends it to Alice.

(iii) Alice computes

$$w = t - c(x - X),$$

and $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$. Alice sends w to Bob.

(iv) Bob checks $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$ and

$$g^{w-cX} T^c =?\ d \pmod{n}.$$

If the equation holds, Alice proves knowledge of the discrete logarithm of T that lies in the range $[X - 2^{\alpha(l+l_c)+1}, X + 2^{\alpha(l+l_c)+1}]$.

Remark 3.3.3. The asymmetry of this proof — that Alice must know a value in a more restrictive range that Bob will be convinced of — seems unusual at first, but the following proof will illustrate this is sufficient to prove the properties needed.

Proof: The proof for correctness is straightforward. A prover with the knowledge of discrete logarithm can always convince a verifier.

The protocol can be proved statistical honest-verifier zero-knowledge for $\alpha > 1$. A simulator for protocol transcripts between the honest prover and the honest verifier works as follows: it selects $\hat{c} \in_R \{0,1\}^{l_c}$, and $\hat{w} \in_R \pm\{0,1\}^{\alpha(l+l_c)+1}$, and computes:

$$\hat{d} = g^{\hat{c}} T^{\hat{w} - \hat{c}X} \pmod{n}.$$

The simulator outputs the transcript $(T, \hat{d}, \hat{c})$.

It is needed to show that the simulated transcripts are statistically indistinguishable from the transcripts that are generated between the honest prover and the honest verifier. With the consideration that an honest prover uniformly selects $t \in_R \{0,1\}^{\alpha(l+l_c)}$, the probability distribution $P(w)$ can be calculated as follows.

(i) For $-X + 2^{\alpha(l+l_c)} < w < X - 2^{\alpha(l+l_c)}$ for each of the 2^{l_c} different c, there is a t such that $w = t - c(x - X)$. Thus, $P(w) = 2^{l_c}/(2^{l_c}2^{\alpha(l+l_c)+1}) = 1/2^{\alpha(l+l_c)+1}$.

(ii) For $w \geq X + 2^{\alpha(l+l_c)}$ or $w \leq -X - 2^{\alpha(l+l_c)}$, since it is impossible to solve the equation $w = t - c(x - X)$, $P_1(w) = 0$.

(iii) For the other selection of $w \in Z$, $P(w) \in [0, 1/2^{\alpha(l+l_c)+1})$.

The simulator's probability distribution of $\hat{w}$, $P'(\hat{w})$, is $1/2^{\alpha(l+l_c)+1}$ for $\hat{w} \in \pm\{0,1\}^{\alpha(l+l_c)}$, and 0 for other situation.

Therefore, the absolute difference between $P'(\hat{w})$ and $P(w)$ is 0 for case 1 and 2. For case 3, $P'(\hat{w})$ is either 0 or $1/2^{\alpha(l+l_c)+1}$. In the worst case the absolute difference between

$P'(\hat{w})$ and $P(w)$ is $1/2^{\alpha(l+l_c)+1}$. The number of possibilities for w in case 3 is 2^{l+l_c+2}, thus the statistical distance of $P'(\hat{w})$ and $P(w)$ is at most

$$2^{l+l_c+2}/2^{\alpha(l+l_c)+1} = 1/2^{(\alpha-1)(l+l_c)-1}.$$

Due to the choice of l, l_c, α (Note: $\alpha > 1$), the probability difference between $P(w)$ and $P'(\hat{w})$ is statistically insignificant.

To prove the protocol is a proof of knowledge, it is necessary to show that a knowledge extractor is able to recover the group certificate when it has found two accepting tuples under the same commitment and different challenges from a verifier. Let (d, w, c) and (d', w', c') be two accepting tuples.

Since $d \equiv g^{w-cX}T^c \equiv g^{w'-c'X}T^{c'} \pmod{n}$,

$$g^{w'-w+(c-c')X} \equiv T^{c-c'} \pmod{n}.$$

By Lemma 3.2.4, $c-c'$ must divide $w'-w+(c-c')X$. Otherwise an instance of the flexible RSA problem can be solved, i.e., $g \equiv v^e \pmod{n}$ for some v, and $1 < e < (c-c')$, which contradicts the strong RSA assumption (Assumption 3.2.1).

Therefore, a knowledge extractor obtains

$$x = X + (w'-w)/(c-c').$$

Since $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$, x must be in the range of $[X - 2^{\alpha(l+l_c)+1}, X + 2^{\alpha(l+l_c)+1}]$.

$\square$

CHAPTER 4

AN EFFICIENT PUBLIC KEY CRYPTOSYSTEM FOR EMBEDDED SYSTEMS

This chapter presents an efficient public key cryptosystem for embedded systems, which includes protocols for authentication, authenticated key exchange, encryption and revocation [1]. The new system is proved secure under the strong RSA assumption, and the computational Diffie-Hellman assumption.

4.1. Introduction

4.1.1. Background

Recall the requirements for embedded system scenarios given in Chapter 1: it would be desirable for the devices in the system to use encrypted communication for confidentiality and integrity, and to be able to securely recognize other devices with the same owner. A pair of devices should be able to establish authenticated secure channels on their own, without needing to interact with the owner or any other party at the time, and the system is dynamic, meaning that new devices can enter the system at any time. Therefore, solutions based on symmetric cryptography, whether requiring preloaded keys for all pairs of devices or using an available authority (like in Kerberos), do not satisfy the requirements.

There are many known cryptographic solutions which can solve this problem, including using X.509 certificates issued by the owner, using a group signature or credential system with the owner as the group manager, or using identity-based encryption. However, the scenario has some unique properties which allow for a more efficient solution, and the importance of highly optimized protocols in embedded systems has led us to develop such an efficient

[1]Parts of this chapter have been accepted by ITNG06 [31]. Section 4.1, 4.2, 4.3.1, 4.3.2, Theorem 4.4.3, Theorem 4.4.5, Paragraph 1,2,3,4 in section 4.5 use material in [31] with permission under IEEE copyright policy.

solution in this chapter. After defining a model capturing the specific characteristics of embedded systems in the next section, a discuss will be given on how various aspects of the model allow improvement upon these existing techniques.

4.1.2. The Model

This section defines a model which captures security requirements in a system with three key properties: a small set of devices (typically a few devices up through a few thousand sensors), a single completely trusted administrator (the owner of the devices), and no externally meaningful names that need to be linked to keys. To protect communication between members of this set of devices, it is necessary to provide a security protocol to implement authentication and key establishment. The following definition precisely specifies the security requirements of the system. Terminology from the related area of group signatures has been used, but keep in mind an important difference in embedded system: the group members are not independent people, but rather devices ultimately controlled by the single owner. It is also necessary to clarify here that any attacker is assumed to be a probabilistic polynomial time algorithm, so references to things being impossible (such as forging a key) should be understood to mean impossible for a probabilistic polynomial time algorithm.

Definition 4.1.1 (The Model). A group consists a single administrator, and at most several thousand group members. The administrator holds a group master key while each group member holds its group member key. A system supporting authenticated key exchange should satisfy the following properties:

- (Forgery-resistance) A valid group member key can only be produced with knowledge of the group master key, so if only the administrator has access to this key then only the administrator can create valid group member keys.
- (Authentication Soundness) A party can prove membership in the group if and only if it knows a valid group member key.

- (Confidential Key Establishment) Any two members in the group can compute a shared secret that cannot be deduced by outsiders. This shared secret can be used with symmetric cryptography to support confidentiality and integrity of subsequent communication.
- (Robustness) The compromise of a group member will not affect the security of interactions between un-compromised group members. This is in contrast to the shared-key scheme in which the compromise of one group member would reveal all the communication in the system.
- (Forward Secrecy) If the administrator's group master key is obtained, an attacker still can not compromise operations between group members whose keys were established prior to the administrator's compromise. Note that forward secrecy property defined here is a property of authenticated keys, not messages. The more basic property of forward secrecy of messages applies to encryption systems, not key exchange, but it needs to point out that since the key exchange establishes random session keys any larger system using our key exchange protocol will also exhibit forward secrecy of messages.
- (Member Revocation) The administrator can revoke a group member in case it does not belong to the group anymore, or it is broken by an attacker.

The last two properties are optional, whose importance depends on the likelihood of compromise of the administrator or the group members, respectively. If the group master key is kept on a general-purpose computer connected to the Internet, it is more important to consider forward secrecy than in an environment in which the group master key is kept on a dedicated device with highly controlled access. Similarly, in settings such as an unattended sensor network in a hostile environment, it is almost inevitable that group members will be compromised, so revocation is vital. In other settings, such as fixed devices in a physically controlled environment (like appliances in a home) the chances of member compromise is low, so revocation is less important.

4.1.3. The Result and Related Works

This chapter proposes a simple and efficient authentication scheme which can be deployed in applications which satisfy the above model. Its security is based on the strong RSA assumption and the computational Diffie-Hellman assumption. This section shows how the proposed scheme is related to other solutions for this problem, and describe which properties of the proposed system enable us to improve upon these other solutions.

X.509 certificates [39] provide an obvious solution for authentication, which is what SSL does for connections over the Internet. While this does provide the group member authentication needed, the primary purpose of X.509 certificates is to bind a meaningful identity and other properties to a cryptographic key, which is unnecessary in our setting. To support the flexible requirements of X.509 certificates, a typical certificate would be a separate object that is several times as large as the key that it is authenticating. For example, the current X.509 certificate authenticating the 128 byte (i.e., 1024 bit) key for www.amazon.com is 945 bytes long, so requires 738% overhead. By contrast, our authentication is implicit in the key itself, so requires no overhead at all (although to attain forward secrecy it is necessary to double the size of the key). Furthermore, a device which wants to authenticate itself using an X.509 certificate must perform an RSA signature, using a private exponent that is as long as the modulus, so an RSA signature with a 1024 bit key would require an average of 1536 modular multiplications. By contrast, the proposed scheme uses a short (160-bit) private exponent, allowing a device with a 1024 bit key to authenticate itself with an average of 240 modular multiplications, roughly 15% of the number required by the RSA signature.

Identity-based encryption (IBE) [54, 11] can also be used to solve basic problems in embedded system, but IBE is somewhat less efficient than the proposed construction in this chapter. More important than efficiency in this case is the inability of IBE to support forward secrecy. Since the basis of IBE is the ability for the key manager to compute a private key from the corresponding public key, if the group manager is compromised then all group member keys are immediately compromised.

Self-certified public keys [34, 49], certificate-less public keys [1], and group signature schemes [2] also provide the capabilities needed. However, the basic model for these systems includes independent parties as the authenticated entities, and they do not trust the group manager. In an embedded system, the group members are passive devices that are owned by, and hence completely trust, the group manager. The effort that these techniques use to protect against a malicious group manager is unnecessary in target scenarios, since everyone is on the same team. In addition, properties such as anonymity and unlinkability, as supported by group signatures, have no meaning in embedded systems, so are unnecessary complications.

The proposed system has a few additional properties which make it especially appropriate for computation and memory constrained devices. First, while based on modular exponentiation like RSA, short private exponents can be safely used, greatly improving the complexity of authentication. While many parts of the proposed system map to standard RSA, the RSA public key is hidden in the proposed system, so attacks on RSA with a small decryption exponent [59, 10] do not work against our system. Second, keys are bound to short, unique tags, that are typically 16 or 24 bits. Hence, if revocation needs to be supported, it can be done through a very compact revocation list. In addition, an efficient secure re-keying scheme is also proposed for highly dynamic environments.

4.2. The System Setup

The group administrator sets various parameters, the lengths of which depend on a *security parameter*, denoted by σ.

4.2.1. Parameter Setting of the Group Administrator

A single administrator generates group member keys for the system, based on some public and private values held by the administrator. The administrator's values are:

- n, g: n is a special RSA modulus such that $n = pq$, $p = 2p' + 1$, and $q = 2q' + 1$, where p and q are each at least σ bits long (so $p, q > 2^{\sigma}$), and p' and q' are prime. g is a generator of the cyclic group QR_n. n and g are public values while p and q are kept secret by the administrator.
- l_s: l_s is the length of a group member private key, with the restriction that $l_s < \sigma - 1$ so that all member private keys are guaranteed to be less than $\min(p', q')$. l_s also needs to be large enough where a brute force search is not feasible.
- l_t: l_t is the length of the group member public key identifiers/tags, with the restriction that $l_t < l_s$. The only lower bound on this parameter is that it must be large enough so that every group member can be assigned a distinct prime number of length l_t. Some typical examples for this length might be 8 (for groups of up to 23 members), 16 (for groups of up to 3030 members), 24 (for groups of up to 513,708 members), or 32 (for groups of up to 98,182,656 members).
- One-way hash function: $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{l_s}$.

Based on the speed of current number theoretic algorithms and cryptanalytic attacks using current hardware, for moderate sized groups of up to a few thousand members, settings of $\sigma = 512$ (so n is 1024 bits), $l_s = 160$, $l_t = 24$ offer strong security.

4.2.2. Generation of the Group Member Key

The method for the creation of a group member key is straightforward. The administrator picks two random prime numbers s and t with lengths l_s and l_t, respectively, where t has not been previously used for a different group member. The administrator computes

$$E = g^{s^{-1}t^{-1}} \pmod{n},$$

where s^{-1}, t^{-1} are the inverses of s, t modulo $|QR_n| = p'q'$, respectively. Note that since $l_t < l_s < \sigma - 1$, both s and t are smaller then $min(p', q')$, so are relatively prime to $p'q'$, which guarantees that these inverses exist. s is the private part of a group member key, while

(E, t) is the public part. t is the unique identifier, or tag, to represent a group member key. To some degree, the tag t can be seen as a special identity as in self-certified public key schemes, or certificate-less public key schemes [34, 1]. However, in practice, a prime number is not treated as a meaningful identity such as email address, etc.

After the administrator delivers the group key to a group member through a secure method, it stores (E, t) in its database and destroys private key s. Therefore, even if the administrator is compromised, the private key for each group member remains secure.

4.3. Authentication Protocols

This section presents the authentication scheme, the authenticated key exchange, and the encryption scheme.

4.3.1. Authentication Protocol

Suppose a group member Alice needs to authenticate herself to another party Bob (who may or may not also be a group member). The authentication protocol works as follows.

(i) Alice sends (E, t) to Bob.

(ii) If the bit length of t is correct, Bob picks a random integer $r \in_R \{2, \ldots, n-1\}$, computes

$$E' = E^{tr} \pmod n \quad \text{and} \quad W = \mathcal{H}(g^r \pmod n),$$

and sends E', W to Alice. Otherwise, Bob aborts the authentication.

(iii) Alice computes

$$E'' = E'^s \pmod n.$$

If $\mathcal{H}(E'') = W$, then Alice sends E'' to Bob; otherwise, she finds that Bob was cheating, and aborts the protocol.

(iv) Bob checks

$$E'' =?\ g^r \pmod n. \tag{2}$$

If equation (2) holds, this finishes the authentication of Alice to Bob.

The purpose of W is for Bob to prove that he knows the expected answer $g^r \pmod{n}$, without giving away the answer. Since Bob knows $g^r \pmod{n}$ already, when Alice responds with $g^r \pmod{n}$ it's clear that Bob learns no new information. If $\mathcal{H}$ is a true one-way hash function, then that makes the authentication protocol "zero-knowledge," and prevents chosen message attacks.

4.3.2. Mutual Authentication and Key Exchange

Based on the authentication protocol, Alice and Bob can complete a mutual authentication procedure to implement session key generation. The procedure actually implements an authenticated Diffie-Hellman key exchange [26]. It works as follows.

(i) Alice and Bob exchange their public keys: (E_a, t_a), and (E_b, t_b).

(ii) Alice and Bob pick random numbers r_a, r_b, calculate challenges $C_a = E_b^{t_b r_a}$, $C_b = E_a^{t_a r_b}$ and exchange challenges.

(iii) Alice calculates $C_b^{s_a}$ which is g^{r_b} if all parties follow the protocols. Then Alice can get $g^{r_a r_b}$ which Bob is also supposed to obtain. Alice and Bob use this common value to derive a session key.

(iv) Key confirmation step: Alice uses the session key to encrypt g^{r_b} using a symmetric encryption algorithm. Bob uses the session key to encrypt g^{r_a} using same method. Alice and Bob exchange the confirmation message. If Alice/Bob find the decrypted value is equal to their own result, this finishes the key exchange protocol. Otherwise, the protocol is aborted.

In a real application, the key confirmation step can be integrated into the subsequent data transmission. Therefore, only two rounds of message exchange are needed for the authenticated key exchange protocol.

4.3.3. Encryption Scheme

An encryption protocol can be extended base on the authentication protocol, which is similar to ElGamal encryption algorithm [27]. It works as follows.

(i) Using Alice's public key (E, t), Bob encrypts a message $m < n$ as a pair (c_1, c_2) such that

$$c_1 = E^{kt} \pmod{n}, \qquad c_2 = Km \pmod{n},$$

where $K = g^k \pmod{n}$ for a random $k \in Z_n$.

(ii) When Alice receives the ciphertext, she decrypts it as

$$K = c_1^s \pmod{n}, \qquad m = c_2 K^{-1} \pmod{n}.$$

It should be noted that this version of encryption is an authenticated one. That is, when Bob sends a ciphertext encrypted by Alice's public key (E, t), he knows only authenticated Alice can decrypt this ciphertext. This is in contrast to the ElGamal encryption scheme in which encryption does not provide authentication.

4.4. Security Properties

This section introduces a slight variant of the Strong RSA Assumption, in which one does not need to find the exponent e itself, but rather find a hidden version of it, g^e. Note that extracting the actual exponent from g^e would require solving the discrete logarithm problem over QR_n.

Assumption 4.4.1 (Hidden Exponent Strong RSA Assumption). Let n be a special RSA modulus, and g be a generator of QR_n. Let $\log_g x$ denote the base g discrete log of x over QR_n, so $g^{\log_g x} \equiv x \pmod{n}$ for all $x \in QR_n$. The *Hidden Exponent Flexible RSA Problem* is the problem of taking a random element $u \in Z_n^*$ and finding a pair (v, w) such that $w \neq g$

and $v^{\log_g w} \equiv u \pmod n$. The *Hidden Exponent Strong RSA Assumption* says that no probabilistic polynomial time algorithm can solve the hidden exponent flexible RSA problem with non-negligible probability.

Note that there is a one-to-one mapping between solutions to the flexible RSA problem and the hidden exponent flexible RSA problem. Among other things, this means that if this variant with a particular fixed hidden exponent can be solved efficiently, then the corresponding "non-hidden" exponent is a weak encryption exponent for RSA. While such exponents do exist (for example, corresponding to small decryption exponents), there is no algorithm that we're aware of that can produce a weak encryption exponent without knowledge of the factorization of n (or, equivalently, the decryption exponent). Note that if one could find such weak encryption exponents efficiently, then one could solve the (standard) flexible RSA problem, so it is unlikely that an attacker could find such a weak exponent to exploit to solve the hidden exponent flexible RSA problem.

Due to the strong RSA assumption, it is obvious that if no pairs (v, e) are known such that $v^e \equiv g \pmod n$, then no one except the administrator can create a valid keypair (E, s, t) such that $E^{st} \equiv g \pmod n$. However, the issue of keypair forgery needs to be addressed when some existing keyparis are known. In practice, more than one sensor could be compromised such that the keypairs are extracted, or some group members collude collude with each other to expose their keypairs. Such a scenario is abstracted as an attack model in which an attacker can collude with a set of legitimate parties, each with a legitimate keypair. A successful attack is one in which a new keypair is generated, with an identifier t that is valid and different from those of the colluding parties. The following theorem shows that, assuming the Strong RSA Assumption, it is intractable for an attacker to forge such a keypair.

Theorem 4.4.2 (Forgery-resistance). *If there exists a probabilistic polynomial time algorithm which takes a list of valid keypairs, $(s_1, E_1, t_1), (s_2, E_2, t_2), \ldots, (s_k, E_k, t_k)$ and with non-negligible probability produces a new keypair (s, E, t) such that $E^{st} \equiv g \pmod{n}$ and $t \neq t_i$ for $1 \leq i \leq k$, then one can solve the flexible RSA problem with non-negligible probability.*

Proof: Suppose there exists a probabilistic polynomial-time algorithm which computes a new legitimate group key based on the available group member keys, and succeeds with some non-negligible probability $p(\sigma)$. Then one can construct an algorithm for solving the flexible RSA problem, given a random input (u, n), as follows (the following makes sense as long as u is a generator of QR_n, which is true with non-negligible probability for random instances — this will be considered more carefully below when analyzing the success probability of the constructed algorithm):

(i) First, check if $\text{GCD}(u, n) = 1$. If it's not, then one of the factors of n is obtained , and a solution is available for the flexible RSA problem. Therefore, $\text{GCD}(u, n) = 1$ is assumed in the following so $u \in Z_n^*$.

(ii) Pick random distinct prime numbers $s_1, s_2, \ldots, s_k$ and $t_1, t_2, \ldots, t_k$ with the required bit lengths, and compute

$$r = s_1 s_2 \ldots s_k t_1 t_2 \ldots t_k,$$

$$g = u^r = u^{s_1 s_2 \ldots s_k t_1 t_2 \ldots t_k} \pmod{n}.$$

Note that since the s_i and t_i values are primes with the appropriate length (constrained to be smaller than the lengths of p' and q'), it must be the case that $\text{GCD}(r, |QR_n|) = 1$, so Property 3.1.2 says that g is a generator of QR_n if and only u is a generator of QR_n.

(iii) Next, create k group member keys, using the s_i and t_i values and E_i values calculated as follows:

$$E_1 = u^{s_2 \ldots s_k t_2 \ldots t_k} \pmod{n}$$

$$
\begin{aligned}
E_2 &= u^{s_1 s_3 \ldots s_k t_1 t_3 \ldots t_k} \pmod n \\
&\vdots \\
E_k &= u^{s_1 s_2 \ldots s_{k-1} t_1 t_2 \ldots t_{k-1}} \pmod n
\end{aligned}
$$

Note that for all $i = 1, \ldots, k$, raising E_i to the power $s_i t_i$ "completes the exponent" in a sense, giving $E_i^{s_i t_i} = u^{s_1 s_2 \cdots s_k t_1 t_2 \cdots t_k} = u^r = g \pmod n$.

(iv) Use the algorithm for creating a new group member key to calculate (s, E, t), where t has the required bit length and $E^{st} = g \pmod n$.

(v) If the forgery algorithm succeeded, then t will be different from all the t_i's, but will have the same length. Therefore, it is impossible for t to be an integer multiple of any of the t_i's, and since the t_i's are prime then it follows that $\mathrm{GCD}(t, t_1 t_2 \cdots t_k) = 1$. Furthermore, since the s_i's are prime and all longer than t, it follows that $\mathrm{GCD}(t, s_1 s_2 \cdots s_k) = 1$, and so $\mathrm{GCD}(t, r) = 1$. Therefore, one can use the Extended GCD algorithm to find a and b such that

$$ar + bt = 1,$$

and let $y = (E^s)^a u^b$. Thus

$$y^t = (E^s)^{at} u^{bt} = u^{ar+bt} = u \pmod n,$$

so the pair (y, t) is a solution to this flexible RSA problem instance.

Let's now analyze the probability that the above algorithm for solving the flexible RSA problem succeeds. The algorithm succeeds in Step 1 if $\mathrm{GCD}(u, n) \neq 1$, so let P_1 represent the probability of this event, which is negligible. When $\mathrm{GCD}(u, n) = 1$, the algorithm succeeds when the following three conditions are satisfied: (1) $u \in QR_n$, which happens with probability $\frac{1}{4}$, (2) u is a generator of QR_n, which fails for only a negligible fraction of elements of QR_n, due to Property 3.1.1, and (3) the key forgery algorithm succeeds, which happens with probability $p(\sigma)$. Putting this together, the probability that the constructed algorithm succeeds is $P_1 + (1 - P_1)\frac{1}{4}(1 - \mathrm{negl}(\sigma))\, p(\sigma)$, which is non-negligible.

$\square$

The proof only shows that the forgery of a tag t is intractable, without addressing the possibility for the forgery of a new private key under an existing public label t. It is in fact possible to forge a group keypair with a duplicated t as long as the legitimate owner of the key with identifier t cooperates with the attack. Let's see how to achieve such a forgery. For two valid keypairs (E_1, t_1, s_1),and (E_2, t_2, s_2), one can have $E_1^{t_1 s_1} \equiv g \pmod{n}$, and $E_2^{t_2 s_2} \equiv g \pmod{n}$. One can further compute

$$E_1^{t_1 s_1 t_2 s_2} \equiv g^{t_2 s_2} \pmod{n},\ E_2^{t_1 s_1 t_2 s_2} \equiv g^{t_1 s_1} \pmod{n},$$

to obtain

$$(E_1 E_2)^{t_1 t_2 s_1 s_2} \equiv g^{t_1 s_1 + t_2 s_2} \pmod{n}.$$

This equation can be solved to obtain $E'^{t_1 t_2 s_1 s_2} \equiv g \pmod{n}$ by Shamir's lemma (Lemma 3.2.2). This could be viewed as two valid keypairs $(E', t_1, t_2 s_1 s_2)$, or $(E', t_2, t_1 s_1 s_2)$, which can pass the authentication.

However, such an attack is equivalent to key sharing, and if a group member is happy to let someone use its group member key, there is no way to prevent it. In real applications, if a group member key is found to be used by more than one party, it should be assumed being compromised, and the identifier t should be revoked.

As for the question of soundness of the authentication protocol, loosely speaking, under the hidden exponent strong RSA assumption, it can be proved that only parties with a legitimate keypair (provided by the administrator) can succeed in the authentication protocol. An attacker of the authentication protocol is abstracted as a pair of functions, which provide the two protocol messages the prover needs to provide. Given the public parameters g and n, function PublicKey(g, n) produces a purported public key (E, t) and (optionally) some private information p that it can use later (PublicKey can also use other publicly available information, such as the public keys of legitimate parties). Function Respond(g, n, E, t, p, c) takes the global public parameters g, n, the information E, t, p produced by the PublicKey function, and a challenge c, and produces an answer a to the challenge. Note that an honest verifier creates a challenge as $c = E^{tr}$ for a random r, and accepts the prover's answer if and only if $a = g^r$ which is true if and only if (raising both sides to the $\log_g E^t$ power) $a^{\log_g E^t} = (g^r)^{\log_g E^t} = E^{tr} = c$.

Theorem 4.4.3 (Authentication Soundness). *If there exist probabilistic polynomial time algorithms PublicKey and Respond that succeed with non-negligible probability in fooling the verifier, then there exists a probabilistic polynomial time algorithm that solves the hidden exponent flexible RSA problem with non-negligible probability.*

Proof: Given an input (g, n, u) to the hidden exponent flexible RSA problem, using the attacker's algorithms, one can create a solution as follows:

- Call PublicKey(g, n) and get (E, t, p).
- Call Respond(g, n, E, t, p, u) to get answer a.
- Set $v = a$ and $w = E^t$ and return (v, w) as an answer to the hidden exponent flexible RSA problem.

Note that if the attacker succeeds in providing a valid (E, t) and a, then as noted before the theorem one can have $a^{\log_g E^t} = c$, or using the v and w notation, $v^{\log_g w} = u$. Therefore, (v, w) is a valid solution to the hidden exponent flexible RSA problem if and only if the attacker succeeded in the authentication protocol. Since the latter occurs with non-negligible probability, by the condition of the theorem, the hidden exponent flexible RSA problem is solved with non-negligible probability.

□

To prevent chosen message attacks from a malicious verifier, the verifier is required to send the witness of E'', $\mathcal{H}(E'')$, to Alice. Thus, any malicious attacks from the verifier can be prevented if $\mathcal{H}$ is a true one-way function.

The encryption protocol is an extension of the authentication protocol, and the authenticated key exchange consists of two mutual authentication procedures. Therefore the authenticity property of these additional operations are secure under the same assumptions. However, it is also needed to show the confidentiality for the shared key, or plaintext are also ensured in these two protocols.

Theorem 4.4.4 (Confidential Key Exchange). *Under the computation Diffie-Hellman assumption over QR_n, Alice and Bob create a shared secret among themselves, which can be used to generate a session key.*

Proof: A third party can observe all exchanged messages between Alice and Bob, so breaking the confidentiality of the shared secret between Alice and Bob implies that this third party can calculate $g^{r_a r_b}$ based on $(E_a, t_a, E_b, t_b, C_a, C_b, g)$.

Assume that for random s_a, s_b, r_a, and r_b, there is a probabilistic polynomial time algorithm $\mathcal{A}$ that computes $\mathcal{A}(E_a^{t_a}, E_b^{t_b}, C_a, C_b, g) = g^{r_a r_b}$ with non-negligible probability. Since $E_a^{t_a} = g^{s_a^{-1}}$, $E_b^{t_b} = g^{s_b^{-1}}$, $C_a = g^{s_b^{-1} r_a}$, and $C_b = g^{s_a^{-1} r_b}$, re-writing, this means that

$$\mathcal{A}(g^{s_a^{-1}}, g^{s_b^{-1}}, g^{s_b^{-1} r_a}, g^{s_a^{-1} r_b}, g) = g^{r_a r_b} \tag{3}$$

with non-negligible probability.

Since s_a, s_b, r_a, and r_b are all independent, this means that s_a^{-1}, s_b^{-1}, $s_b^{-1} r_a$, and $s_a^{-1} r_b$ (all of the exponents of the parameters of (3)) are independent, so $\mathcal{A}$ can be viewed as solving the following problem: Given values g^A, g^B, g^C, and g^D, where A, B, C, D are random independent (and unknown) values, $\mathcal{A}(g^A, g^B, g^C, g^D, g) = g^{A^{-1} B^{-1} C D}$ with non-negligible probability.

Now consider a random instance of the computational Diffie-Hellman problem, (g^x, g^y). One can pick random p_1 and p_2, and compute parameters for $\mathcal{A}$ so that one can call $\mathcal{A}(g^{p_1}, g^{p_2}, g^{xp_1}, g^{yp_2}, g)$. Note that all the exponents are random and independent, so with non-negligible probability $\mathcal{A}$ computes $g^{p_1^{-1}p_2^{-1}xy}$. Since p_1 and p_2 are known, this result can be raised to the p_1p_2 power, giving g^{xy}. Therefore, this technique solves random instances of the computational Diffie-Hellman problem with non-negligible probability, contradicting the assumption.

The above argument assumes that s_a, s_b are random and uniformly chosen over all possible exponents, as well as that r_a and r_b are likewise random over the set of possible exponents, and it requires picking p_1 and p_2 which are random over the possible exponents; however, one can't pick uniformly from the exponents since the size of the group QR_n is unknown. This difficulty can be resolved in the same way as the techniques that use groups of unknown order (The protocols for knowledge of a discrete logarithm, knowledge of the equality of two discrete logarithms, etc. See Chapter 3): instead of drawing p_1 and p_2 from the set of possible exponents, they can be drawn from $\{0,1\}^{\alpha \times l_{QR_n}}$, for some $\alpha > 1$, and l_{QR_n} denotes the bit-length of QR_n. The "goodness" of the approximation depends on the value of $\alpha > 1$, which determines the statistical closeness of actual distribution to the ideal one.

$\square$

In the encryption protocol, a plaintext is encrypted by a random key K, and similar arguments as in the above proof can be applied to show confidentiality for this plaintext is achieved under the computational Diffie-Hellman assumption.

Finally, the proposed protocol exhibits a nice forward secrecy property. Specifically, if the administrator key is compromised, an attacker cannot use this knowledge to compute the private keys of the parties who have previously had keys created by the administrator. This is in contrast to recent work in Identity Based Encryption [11] where compromise of the administrator secrets results in compromise of all party's secret keys.

Theorem 4.4.5 (Forward Secrecy). *If the administrator secret (the factorization of n) is discovered at some point, and σ is large enough so that discrete logs are difficult to compute modulo a prime of σ bits, then all past and future key exchanges and authentications by previously authenticated parties are still secure.*

Proof: Recall that during uncompromised behavior, the administrator deletes any private keys that it has access to. It has been proved that solving discrete logarithm problem modulo a composite number is equivalent to factoring the composite number, plus extracting discrete logarithm modulo the factors of the composite number [4]. Once the factorization of n is known, computing a secret key s from the public key (E, t) is basically a discrete logarithm problem (computing the $\log_{E^t} g$ in Z_p and Z_q. Since p and q are each at least σ bits long, and the condition in the theorem requires that computing discrete logs modulo a prime of σ bits is intractable, then operations with previously authenticated keys remain secure even if the factorization of n is known.

□

Note that since each of p and q need to be long enough to support a group with hard discrete logarithm problems, if forward secrecy with a security level comparable to factoring a 1024-bit RSA value is required, then p and q need to be 1024 bits, which makes the modulus 2048 bits.

The property of forward secrecy makes the new construction more robust than the identity based public key scheme in which compromsing the administrator means the whole system is broken immediately. It should be pointed out if the administrator is compromised, an attacker can create new legitimate group member keys. Therefore, forward secrecy can not prevent any active attack.

4.5. Group Member Key Revocation

In case a group member does not belong to the group anymore, or it is compromised by an attacker, the administrator should notify all the group members that a party holding a certain member key can not be regarded as a legitimate group member. In X.509 certificate based public key cryptosystems, the revocation is implemented by a "revocation list" which contains all the revoked certificates. This is a "black list" technique. In the proposed scheme, a group member is revoked by its tag, which is typically only 2 or 3 bytes, so revocation lists are quite compact. Therefore, operations on the revocation list will use less system resources.

In dynamic settings, with many membership changes, revocation lists can become large and cumbersome even with short identifiers. To solve this problem, this section introduces an efficient method for re-keying all group members. In this re-keying technique, only group member public keys change, while the members retain their existing private keys without having to communicate them in any form whatsoever to the administrator. Thus, this operation is significantly less sensitive than the original key establishment step.

The re-keying techniques works as follows:

(i) When a group member leaves the group, or is found to be broken by an attacker, or the administrator needs to update all the group member keys following certain security policies, the administrator picks a random value $r \in \{2, \ldots, |QR_n|\}$ with $\mathrm{GCD}(r, |QR_n|) = 1$, and then computes

$$h = g^r \pmod{n}.$$

h will be used as a new generator for QR_n.

(ii) For each group member that the administrator wants to retain, the administrator computes the new public key based on the existing public key in its database as follows:

$$E' = E^r = (g^{s^{-1}t^{-1}})^r = (g^r)^{s^{-1}t^{-1}} = h^{s^{-1}t^{-1}} \pmod{n}.$$

(iii) The administrator publishes the new generator h and the new public key for each group member. Later, all the group members can retrieve their new public keys.

This mechanism does not require any computation by the group member. All the computation overhead is on the administrator's side. For a small group with at most a few thousand group members, it is a simple task. Furthermore, the administrator can pre-compute all the new public keys, and only publish those are still valid. Therefore, the administrator can update all the group member keys in nearly real time. At the same time, each group member still keeps its private key and unique identifier. This technique can be combined with a revocation list for an efficient combined black-list/white-list group management system.

For an excluded group member, with existing keypair (E, s, t) that uses generator g over QR_n, he has to solve the "re-keying problem" to obtain a new valid key: computing $E' = h^{s^{-1}t^{-1}} = g^{rs^{-1}t^{-1}} \pmod{n}$ based on $g^{s^{-1}t^{-1}} \pmod{n}$ and $h = g^r \pmod{n}$ without knowing r, t^{-1} or s^{-1}. The following theorem shows that re-keying problem is intractable due to the strong RSA assumption.

Theorem 4.5.1. *If there exists an algorithm that can compute an updated group member public key without knowledge of the administrator's secret value, then there exists an algorithm that solves the flexible RSA problem with non-negligible probability.*

Proof: Suppose there exist a probabilistic polynomial time algorithm $\mathcal{A}$ to solve the re-keying problem with non-negligible probability. That is, for an input (E, s, t, g, h) such that $E^{st} \equiv g \pmod{n}$, and $h \equiv g^r \pmod{n}$ for an unknown random r, $\mathcal{A}$ outputs a E' such that $E'^{st} \equiv h \pmod{n}$.

Let's see how to solve a flexible RSA problem for a random instance (u, n) with non-negligible probability. One picks a random $v \in_R QR_n$, which can be done by squaring a random element in Z_n, and two random prime numbers s, t with the bit-length of l_s, and l_t, respectively. Then one can have

$$g = v^{st} \pmod{n}.$$

Due to the property 3.1.1, the probability of g not being a generator of QR_n, denoted by $\text{negl}(\sigma)$, is negligible. Since n is a special RSA modulus, the probability of $u \in QR_n$ is $\frac{1}{4}$.

Therefore $u \equiv g^r \pmod{n}$ for some unknown r with the probability of $\frac{1}{4}(1 - \mathsf{negl}(\sigma))$. Now one can call the algorithm $\mathcal{A}(v, s, t, g, u)$ for the "re-keying problem" to compute a v' such that

$$v'^{st} \equiv u \pmod{n}.$$

Thus an instance of flexible RSA problem has been solved with the probability of $\frac{1}{4}(1 - \mathsf{negl}(\sigma))$, which is a non-negligible probability. However, this is impossible under the strong RSA assumption. Therefore, no probabilistic polynomial time algorithm $\mathcal{A}$ exists to solve the re-keying problem. □

4.6. Summary

this chapter presents an efficient public key cryptosystem, which includes protocols for authentication, authenticated key exchange and encryption, for applications of embedded systems with a single trusted manager. The construction has been proved secure under the strong RSA assumption, and the computational Diffie-Hellman assumptions. Finally, efficient methods for key revocation and re-keying have been proposed.

CHAPTER 5

GROUP SIGNATURE WITH VARIABLE LINKABILITY

The rest of this dissertation will introduce the results on anonymous authentication. This chapter presents an efficient group signature scheme that supports signature claiming and variable linkability. Based on this scheme, an efficient construction is furtherly proposed for for direct anonymous attestation in a Trusted Computing Platform [1]. These two schemes are based on structures similar to the public key scheme in the previous chapter. The following chapter presents a general construction to carry out anonymous authentication in additional environments.

5.1. Introduction

Chaper 1 reviewed available techniques for anonymous authentication, i.e., group signature, traceable signature and direct anonymous attestation. Numerous constructions with different features have been proposed to accommodate different settings. This raised the question which will be addressed in this chapter: Can we devise a construction which combines features from different authentication primitives? More specifically, can we have a group signature scheme which also supports signature claiming and variable anonymity?

To understand one problem with signature claiming, consider that many group signature schemes work by having the signer include their identity in the signature, encrypted using a semantically secure encryption algorithm and the open authority's public key. For example, consider a system in which the open authority uses El Gamal encryption over a cyclic group with generator g that provides semantic security (such as a subgroup of Z_p^* of prime order), where the open authority has public key y. Then the signature created by a party with identity

[1]Parts of this chapter have been accepted by IPCCC2006 [32]. Section 5.1, 5.2, 5.3 use materials in [32] with permission under IEEE copyright policy.

I could include the El Gamal encryption of the identity I using ciphertext (Iy^r, g^r), where r is a random exponent. The open authority can clearly open this to reveal the identity if necessary. Furthermore, if the signer keeps a record of all the random r values that she used, then a signature can be claimed by simply revealing the r value so that any user can decrypt the identity. However, keeping the complete list of all used r values is both inefficient and a security risk, so the challenge is to support signature claiming without requiring the group member to keep a record of its random values.

As for the variable anonymity property, it is particularly important when group signatures are used for anonymous authentication, since it is the only way key sharing violations can be detected. More specifically, while the standard group signature scheme can use the open authority to identify a user that performs malicious actions, consider what happens when one authorized user shares his authentication credential with a set of co-conspirators. In the scenario for a pay web site, a large set of users could share a single subscription. Since all authentications are completely unlinkable in a group signature scheme, it would be impossible to determine whether 1000 requests coming in during a day are from 1000 different valid users or from 1000 people sharing a single valid credential. Introducing linkability for a limited time period is the only way to detect this, and if an unusually high number of requests using the same credential come in from different IP addresses during the same day, then this could be flagged as potentially malicious behavior, and the open authority could then open the signatures to determine the real owner of this credential for further investigation.

This chapter presents a construction for a group signature scheme that supports both signature claiming and variable linkability. The proposed construction is built up from the group signature scheme due to Camenisch and Michels [17], which will be referred to as the CM scheme. The added capabilities do not adversely affect the time complexity of the various operations, so we preserve the efficiency of the CM scheme while providing these additional features.

A method has also been presented to convert the proposed group signature scheme into the construction that implements direct anonymous attestation in a Trusted Computing Platform. The new scheme is much more efficient in the Sign and Verify protocols than the current solution.

5.2. The Model

This section introduces the model for the proposed signature scheme, which is a variant of the group signature model (e.g. [2]). Both these two models support procedures Setup, Join, Sign, Verify, and Open. The proposed signature scheme supports linkability identifiers in the sign protocol and supports additional procedures Claim (Self-trace) and Claim-Verify.

Definition 5.2.1. A group signature scheme with signature claiming and variable linkability is a digital signature scheme with three types of participants: A group manager, an open authority, and group members. It consists of the following procedures:

- **Setup**: For a given security parameter σ, the group manager produces system-wide public parameters and a group manager master key for group membership certificate generation.
- **Join**: An interactive protocol between a user and the group manager. The user obtains a group membership certificate to become a group member. The public certificate and the user's identity information are stored by the group manager in a database for future use.
- **Sign**: Using his group membership certificate and his private key, a group member creates an anonymous group signature for a message. The signature incorporates a linkability class identifier, LCID, which can be negotiated between the signer and the verifier in an interactive setting.
- **Verify**: A signature is verified to make sure it originates from a legitimate group member without knowledge of which particular one.

- **Open**: Given a valid signature, an open authority discloses the underlying group membership certificate.
- **Claim (Self-trace)**: A group member creates a proof that he created a particular signature.
- **Claim_Verify**: A party verifies the correctness of the claiming transcript.

Similar to a group signature, the proposed signature scheme should satisfy the following properties:

- **Correctness**: Any valid signature can be correctly verified by the Verify protocol and a valid claiming proof can be correctly verified.
- **Forgery-Resistance**: A valid group membership certificate can only be created by a user and the group manager through Join protocol.
- **Anonymity**: It is infeasible to identify the real signer of a signature except by the open authority or if the signature has been claimed.
- **Unlinkability**: It is infeasible to link two different signatures made by the same group member if they incorporate different LCIDs.
- **Non-framing**: No one (including the group manager) can sign a message in such a way that it appears to come from another user if it is opened.
- **Non-appropriation**: No one (including the group manager) can make a valid claim for a signature which they did not create.

5.3. A Group Signature Scheme with Signature Claiming and Variable Linkability

This section describes the implementation of a group signature scheme that supports signature claiming and variable linkability. As mentioned earlier, this scheme is an enhanced version of the Camenisch and Michels group signature scheme [17], the CM scheme.

5.3.1. System Parameter Setting

The group manager picks a security parameter σ, and generates the system parameters as follows:

- n, g, h: n is a special RSA modulus such that $n = pq$, where p and q are each at least σ bits long (so $p, q > 2^{\sigma}$), and $p = 2p' + 1$, and $q = 2q' + 1$, with p' and q' both being prime. g, h are random generators of the cyclic group QR_n. n, g, h are public values while p and q are kept secret by the administrator. The bit-length of the order of QR_n, l_g, is also publicly available.
- α, l_c, l_s: Security parameters that are greater than 1.
- X: A constant integer, $2^{\alpha(l_s+l_c)+2} < X < n$.
- Two strong collision-resistant hash functions: $\mathcal{H}_1 : \{0,1\}^* \rightarrow Z_n^*$, and $\mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_c}$.

An illustration of the system parameters is the setting of $\sigma = 1024$ (so n is 2048 bits), $l_g = 2046$, $\alpha = 9/8$, $X = 2^{860}$, $l_s = 600$ and $l_c = 160$.

The open authority creates his ElGamal public keypair [27], i.e., a random private key x and corresponding public key y such that $y = h^x \mod n$. Note that the open authority may or may not be the same as the group manager.

5.3.2. Join Protocol

The same Join protocol is adopted as in the CM scheme. A group member obtains its group membership certificate as a keypair (E, s), such that s is prime, $s \in [X, X + 2^{l_s}]$, and

$$E^s \equiv g \pmod{n}.$$

s is the group member's private key and is kept secret by the group member. The Join protocol works as follows.

A user sends $T_1 = ss', T_2 = g^s \pmod{n}, T_3 = g^{s'} \pmod{n}$ to the group manager, where s' is another prime number. The choice of s, s' should make it intractable to factor T_1. While the special form of s makes this factorization problem easier than general factorization, the suggested parameter settings are secure based on the work of Coppersmith, who has investigated the factorization issue when the high bits are known for the factors of an integer [24]. Readers may refer to the CM scheme paper for more details.

The user proves to the group manager that

- T_1 is the product of two prime numbers by the protocol in [33].
- The discrete logarithms of T_2 with respect to g, and g^{T_1} with respect to T_3, are equal, and lies in the interval $[X - 2^{\alpha(l_s+lc)+1}, X + 2^{\alpha(l_s+lc)+1}]$.

After the group manager is convinced that (T_1, T_2, T_3) are correctly formed, it computes

$$E = T_3^{T_1^{-1}} = (g^{s'})^{(ss')^{-1}} = g^{s^{-1}} \pmod{n},$$

and sends E to the user.

5.3.3. Variable Anonymity Parameter

To achieve variable anonymity, each signature will belong to a "linkability class" that is identified using a "linkability class identifier," or LCID. All signatures made by the same group member with the same LCID are linkable, and in an interactive authentication protocol the LCID can be negotiated and determined by the two parties. For example, to link authentications to a single server over a single day, the LCID could simply be the server name concatenated with the date. If the same LCID is always used with a particular server (e.g., the server name), then the result is a pseudo-anonymity system. If complete anonymity is desired, the signer can simply pick a random LCID (which is possible if the server isn't concerned with linkability and allows arbitrary LCIDs).

5.3.4. Sign Protocol

(i) Derive a random generator j of QR_n by hashing the LCID of this signature.

$$j = (\mathcal{H}_1(LCID))^2 \pmod{n}.$$

In the random oracle model, for different LCIDs, different and independent j values will be computed. Due to Property 3.1.1, the probability that j is not a generator of QR_n is negligible.

(ii) Generate a random blinding integer $b \in_R \{0,1\}^{l_g}$ and compute:

$$T_1 = Ey^b \pmod{n},\ T_2 = h^b \pmod{n},$$

$$T_3 = j^s \pmod{n}.$$

(iii) Randomly choose $t_1 \in_R \{0,1\}^{\alpha(l_s+l_c)}$, $t_2 \in_R \{0,1\}^{\alpha(l_g+l_s+l_c)}$, and $t_3 \in_R \{0,1\}^{\alpha(l_s+l_c)}$.

- Compute (all computations done modulo n) $d_1 = T_1^{t_1}/y^{t_2}$, $d_2 = T_2^{t_1}/h^{t_2}$, and $d_3 = j^{t_1}$;
- $c = \mathcal{H}_2(g||h||y||j||T_1||T_2||T_3||d_1||d_2||d_3||m)$, where m is a message to be signed.
- $w_1 = t_1 - c(s - X)$, $w_2 = t_2 - csb$.

(iv) Output the signature $(c, w_1, w_2, T_1, T_2, T_3)$ on message m.

Remark 5.3.1. Note that the main difference between the new method and the CM scheme is the computed value T_3 — the corresponding value in the CM scheme (denoted d in their notation) is computed as $h^s j^b$, where j is a fixed generator in their scheme.

5.3.5. Verify Protocol

(i) Compute the same generator j, and

$$c' = \mathcal{H}_2(g||h||y||j||T_1||T_2||T_3||g^c T_1^{w_1-cX}/y^{w_2}||$$

$$T_2^{w_1-cX}/h^{w_2}||j^{w_1-cX}T_3^c||m)$$

(ii) Accept the signature if and only if $c = c'$ and $w_1 \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$, $w_2 \in \pm\{0,1\}^{\alpha(l_g+l_s+l_c)+1}$.

5.3.6. Claim and Claim_Verify

A group member uses the protocol of knowledge of a discrete logarithm in an interval (Protocol 3.3.3) to claim his signature by proving knowledge of the discrete logarithm s of T_3 w.r.t. base j, and proving that s lies in the range of valid private keys.

5.3.7. Open Protocol

For a valid signature, an open authority can open the signature to find its originator as follows

$$E = T_1/T_2^x \pmod{n}.$$

For the non-framing property, the open authority must also issue a proof that it correctly revealed the group member. That is, the open authority proves that the discrete logarithms of y, and $T_1/E \pmod{n}$ with respect to g and T_2, respectively, are equal through Protocol 3.3.2.

5.3.8. Security Properties

The new scheme uses the same certificate as in the CM scheme, so properties that depend only on the form of the membership certificate such as forgery resistance and non-framing are unaffected by these changes.

The following lemma deals with the valid range of system parameters.

Lemma 5.3.2. *If* $X > 2^{\alpha(l_s+l_c)+2}$, $\alpha, l_s, l_c > 1$, *then* $(X - 2^{\alpha(l_s+l_c)+1})^2 > X + 2^{\alpha(l_s+l_c)+1}$.

Proof:

$$(X - 2^{\alpha(l_s+l_c)+1})^2 - (X + 2^{\alpha(l_s+l_c)+1}) = X^2 - X2^{\alpha(l_s+l_c)+2} + 2^{2\alpha(l_s+l_c)+2} - X - 2^{\alpha(l_s+l_c)+1}$$

$$= X(X - 2^{\alpha(l_s+l_c)+2} - 1) + 2^{2\alpha(l_s+l_c)+2} - 2^{\alpha(l_s+l_c)+1}$$

Since $\alpha, l_s, l_c > 1$, and $X > 2^{\alpha(l_s+l_c)+2}$, the equation is greater than 0.

$\square$

The following theorem deals with the security of group membership certificate, which is directly from the CM scheme.

Theorem 5.3.3. *Under the strong RSA assumption, only the group manager with knowledge of the factors of n can compute a legitimate keypair* (E, s) *such that* $E^s = g \pmod{n}$*, and* $s \in [x - 2^{\alpha(l_s+l_c+1)}, x + 2^{\alpha(l_s+l_c+1)}]$ *through Join protocol.*

The issue of keypair forgery must be addressed in case an attacker can obtain a set of legitimate keypairs. A successful attack is one in which a new keypair is generated that is valid and different from current keypairs. The following theorem shows that, assuming the Strong RSA Assumption, it is intractable for an attacker to forge such a keypair.

Theorem 5.3.4 (Forgery-resistance). *If there exists a probabilistic polynomial time algorithm which takes a list of valid keypairs,* $(E_1, s_1), (E_2, s_2), \ldots, (E_k, s_k)$ *and with non-negligible probability produces a new valid keypair* (E, s) *such that* $E^s \equiv g \pmod{n}$ *and* $s \neq s_i$ *for* $1 \leq i \leq k$*, then one can solve the flexible RSA problem with non-negligible probability.*

Proof: Suppose there exists a probabilistic polynomial-time algorithm which computes a new legitimate keypair based on the available keypairs, and succeeds with some non-negligible probability $p(\sigma)$. Then one can construct an algorithm for solving the flexible RSA problem, given a random input (u, n), as follows (the following makes sense as long as u is a generator of QR_n, which is true with non-negligible probability for random instances — This will be considered more carefully below when analyzing the success probability of the constructed algorithm):

(i) First, check if $\text{GCD}(u, n) = 1$. If it's not, one of the factors of n can be obtained, and a solution can be easily calculated for the flexible RSA problem. Therefore, $\text{GCD}(u, n) = 1$ is assumed in the following, so $u \in Z_n^*$.

(ii) Pick random prime numbers $s_1, s_2, \ldots, s_k$ in the required range $[X - 2^{l_s}, X + 2^{l_s}]$, and compute

$$r = s_1 s_2 \ldots s_k,$$

$$g = u^r = u^{s_1 s_2 \ldots s_k} \, (\text{mod } n).$$

Note that since the s_i values are primes strictly less than either p' or q', it must be the case that $\text{GCD}(r, |QR_n|) = 1$, so Property 3.1.2 says that g is a generator of QR_n if and only u is a generator of QR_n.

(iii) Next, create k group keypairs, using the s_i values and E_i values calculated as follows:

$$\begin{aligned} E_1 &= u^{s_2 \ldots s_k} \, (\text{mod } n) \\ E_2 &= u^{s_1 s_3 \ldots s_k} \, (\text{mod } n) \\ &\vdots \\ E_k &= u^{s_1 s_2 \ldots s_{k-1}} \, (\text{mod } n) \end{aligned}$$

Note that for all $i = 1, \ldots, k$, raising E_i to the power s_i "completes the exponent" in a sense, giving $E_i^{s_i} = u^{s_1 s_2 \cdots s_k} = u^r = g \, (\text{mod } n)$.

(iv) Use the assumed forgery algorithm for creating a new valid keypair (E, s), where $s \in [X - 2^{\alpha(l_s + l_c)+1}, X + 2^{\alpha(l_s + l_c)+1}]$, and $E^s = g = u^r \ (\text{mod } n)$.

(v) If the forgery algorithm succeeds, then s will be different from all the s_i's. By Lemma 5.3.2, s cannot be the product of s_i, s_j, $1 \leq i, j \leq k$. Therefore, either $\text{GCD}(s, s_1 s_2 \cdots s_k) = 1$, or $GCD(s, s_1 s_2 \cdots s_k) = s_i$, $1 \leq i \leq k$. In the first case, due to Lemma 3.2.2, one can find a pair (y, s) such that

$$y^s = u \ (\text{mod } n)$$

so the pair (y, s) is a solution to the flexible RSA problem instance.

In the second case, assume $s = v \times s_i$, then $v < X - 2^{\alpha(l_s+l_c)+1}$, and $GCD(v, s_1 s_2 \cdots s_k) = 1$ (or GCD(v,r) = 1). It can be obtained that

$$E^s \equiv E^{v s_i} \equiv (E^{s_i})^v \equiv u^r \pmod{n}.$$

Again by Lemma 3.2.2, a pair (y, v) can be found such that

$$y^v = u \pmod{n}.$$

so the pair (y, v) is a solution to the flexible RSA problem instance.

Let's analyze the probability that the above algorithm for solving the flexible RSA problem succeeds. The algorithm succeeds in Step 1 if GCD$(u, n) \neq 1$, so let P_1 represent the probability of this event, which is negligible. When GCD$(u, n) = 1$, the algorithm succeeds when the following three conditions are satisfied: (1) $u \in QR_n$, which happens with probability $\frac{1}{4}$, (2) u is a generator of QR_n, which fails for only a negligible fraction of elements of QR_n, due to Property 3.1.1, and (3) the key forgery algorithm succeeds, which happens with probability $p(\sigma)$. Putting this together, the probability that the constructed algorithm succeeds is $P_1 + (1 - P_1)\frac{1}{4}(1 - \text{negl}(\sigma))\, p(\sigma)$, which is non-negligible.

□

Now let's address the security of Sign and Verify protocol. The zero-knowledge property of the protocol relies on two aspects: the random choice of the blinding number b, and random generation of a generator j. E is "encrypted" in (T_1, T_2) through the ElGamal encryption algorithm, while s is "hidden" as the discrete logarithm of T_3 for a random generator j produced from a strong collision-resistant hash function $\mathcal{H}_1$. Intuitively, the semantic security property of ElGamal algorithm over QR_n ensures that nothing about E will be revealed by (T_1, T_2). As to s, since j is uniformly distributed over QR_n, and the length of s guarantees that $GCD(s, p'q') = 1$, T_3 is also uniformly distributed over QR_n, and it can be imagined that a simulator can generate this same distribution. Then the zero-knowledge property for

s in the form $T_3 = j^s \pmod{n}$ follows from the decisional Diffie-Hellman assumption over QR_n. These arguments is formalized in the following theorem.

Theorem 5.3.5. *Under the strong RSA assumption, and the decisional Diffie-Hellman assumption over QR_n, the interactive protocol underlying the signature scheme is a statistical honest-verifier zero-knowledge proof of knowledge of a valid group certificate (E, s) such that $E^s = g \pmod{n}$, and s lies in the correct interval.*

Proof: The proof for correctness is straightforward. The protocol is statistical honest-verifier zero-knowledge for $\alpha > 1$. A random oracle can be accessed by all parties for the creation of random generator j. A simulator for protocol transcripts between the honest prover and the honest verifier work as follows: it selects $\hat{c} \in_R \{0,1\}^{l_c}$, and $\hat{w}_1 \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$, $\hat{w}_2 \in \pm\{0,1\}^{\alpha(l_g+l_s+l_c)+1}$,$\gamma \in_R Z_n^*$, and computes:

$$\hat{d}_1 = g^{\hat{c}} T_1^{\hat{w}_1 - \hat{c}X} / y^{\hat{w}_2} \pmod{n}, \ \hat{d}_2 = T_2^{\hat{w}_1 - \hat{c}X} / h^{\hat{w}_2} \pmod{n},$$

$$\hat{j} = j^{\gamma} \pmod{n}, \ \hat{T}_3 = T_3^{\gamma} \pmod{n},$$

$$\hat{d}_3 = \hat{j}^{\hat{w}_1 - \hat{c}X} \hat{T}_3^{\gamma} \pmod{n}.$$

The simulator outputs the transcript $(T_1, T_2, \hat{T}_3, \ \hat{d}_1, \ \hat{d}_2, \ \hat{d}_3, \hat{c}, \ \hat{w}_1, \hat{w}_2)$.

The simulated transcripts are statistically indistinguishable from the transcripts that are generated between the honest prover and the honest verifier. With the consideration that an honest prover uniformly select $t_1 \in_R \{0,1\}^{\alpha(l_s+l_c)}$, $t_2 \in_R \{0,1\}^{\alpha(l_g+l_s+l_c)}$, and $t_3 \in_R \{0,1\}^{\alpha(l_s+l_c)}$. the probability distribution of $P_1(w_1), P_2(w_2)$ can be computed as follows. The probability distribution of $P_1(w_1)$ is analyzed first.

(i) For $-X + 2^{\alpha(l_s+l_c)} < w_1 < X - 2^{\alpha(l_s+l_c)}$ for each of the 2^{l_c} different c, one can find a t_1 such that $w_1 = t_1 - c(s - X)$. Thus, $P_1(w_1) = 2^{l_c}/(2^{l_c} 2^{\alpha(l_s+l_c)+1}) = 1/2^{\alpha(l_s+l_c)+1}$.

(ii) For $w_1 \geq X + 2^{\alpha(l_s+l_c)}$ or $w_1 \leq -X - 2^{\alpha(l_s+l_c)}$, since it is impossible to solve the equation $w_1 = t_1 - c(s - X)$, $P_1(w_1) = 0$.

(iii) For the other selection of $w_1 \in Z$, $P_1(w_1) \in [0, 1/2^{\alpha(l_s+l_c)+1})$.

The simulator's probability distribution of $\hat{w}_1$, $P_1'(\hat{w}_1)$, is $1/2^{\alpha(l_s+l_c)+1}$ for $\hat{w} \in \pm\{0,1\}^{\alpha(l_s+l_c)}$, and 0 for other situation.

Therefore, the absolute difference between $P_1'(\hat{w}_1)$ and $P_1(w_1)$ is 0 for case 1 and 2. For case 3, $P_1'(\hat{w}_1)$ is either 0 or $1/2^{\alpha(l_s+l_c)+1}$. In the worst case the absolute difference between $P_1'(\hat{w}_1)$ and $P_1(w)$ is $1/2^{\alpha(l_s+l_c)+1}$. The number of possibilities for w_1 in case 3 is $2^{l_s+l_c+2}$, thus the statistical distance of $P_1'(\hat{w}_1)$ and $P_1(w_1)$ is at most

$$2^{l_s+l_c+2}/2^{\alpha(l_s+l_c)+1} = 1/2^{(\alpha-1)(l_s+l_c)-1}.$$

Due to choice of l_s, l_c, $P_1'(\hat{w}_1)$ and $P_1(w_1)$ are statistical indistinguishable. Likewise, $P_2'(\hat{w}_2)$ and $P_2(w_2)$ are also statistical indistinguishable.

While T_3 is uniformly distributed over QR_n, so is $\hat{T}_3$. Due to the decisional Diffie-Hellman assumption over QR_n, it is infeasible to distinguish $(j, \hat{j}, T_3, \hat{T}_3)$ from another tuple $(j, \hat{j}, T_3, T_3')$ with $T_3' \in_R QR_n$. Thus, the probability distance of $P_3'(\hat{T}_3)$ and $P_3(T_3)$ is 0.

So far, it is demonstrated that a simulator can successfully to create a transcript which is indistinguishable from a valid transcript in the random oracle model.

Next, it is necessary to show a knowledge extractor is able to recover the group certificate when it has found two accepting tuples under the same commitment and different challenges from a verifier.

Let $(T_1, T_2, T_3, d_1, d_2, d_3, c, w_1, w_2)$ and $(T_1, T_2, T_3, d_1, d_2, d_3, c', w_1', w_2')$ be such tuples.

Since $d_3 \equiv j^{w_1 - cX} T_3^c \equiv j^{w_1' - c'X} T_3^{c'} \pmod{n}$,

$$j^{(w_1' - w_1) + (c - c')X} \equiv T_3^{c-c'} \pmod{n}.$$

By Lemma 3.2.4, under the strong RSA assumption, $c - c'$ has to divide $(w_1' - w_1) + (c - c')X$. Then one can define

$$\tau_1 = (w_1' - w_1)/(c - c') + X. \tag{4}$$

Similarly, $d_2 \equiv T_2^{w_1 - cX}/h^{w_2} \equiv T_2^{w_1' - c'X}/h^{w_2'} \pmod{n}$, so

$$h^{w_2' - w_2} \equiv T_2^{(w_1' - w_1) + (c - c')X} \equiv (T_2^{\tau_1})^{c - c'} \pmod{n},$$

where Lemma 3.2.4 again applies to show that $c - c'$ must divide $(w_2' - w_2)$. Therefore, one can define

$$\tau_2 = (w_2' - w_2)/(c - c'),$$

such that $T_2^{\tau_1} \equiv h^{\tau_2} \pmod{n}$. Applying Lemma 3.2.4 once again to this formula, τ_1 must divide τ_2, so one can compute $\tau_3 = \tau_2/\tau_1$.

Since $d_1 \equiv g^c T_1^{w_1 - cX}/y^{w_2} \equiv g^{c'} T_1^{w_1' - c'X}/y^{w_2'} \pmod{n}$,

$$g^{c - c'} \equiv T_1^{w_1' - w_1 + (c - c')X} y^{w_2 - w_2'} \pmod{n}.$$

Thus

$$g \equiv T_1^{\tau_1}/y^{\tau_2} \equiv T_1^{\tau_1}/y^{\tau_1 \tau_3} \equiv (T_1/y^{\tau_3})^{\tau_1} \pmod{n}.$$

Since $\tau_1 = (w_1' - w_1)/(c - c') + X$, $w_1, w_1' \in \pm\{0,1\}^{\alpha(l_s + l_c)+1}$, and $c, c' \in \{0,1\}^{l_c}$, it follows that $(w_1' - w_1)/(c - c') \in \pm\{0,1\}^{\alpha(l_s + l_c)+1}$. Therefore τ_1 must be in the acceptable range for private key values in the proposed system.

Finally, let $E = T_1/y^{\tau_3} \pmod{n}$ and $s = \tau_1$. We have demonstrated the existence of a knowledge extractor that can find (E, s), such that $E^s = g \pmod{n}$ and s lies in the appropriate range, so (E, s) is a valid membership certificate.

□

Unlinkability follows the same argument as the CM scheme for T_1, T_2. Since a new T_3 is defined in the proposed group signature, it is necessary to show this change still keeps the unlinkability property. Similar to the case in CM scheme, the problem of linking two tuples (j, T_3), (j', T_3') with $j \neq j'$ is equivalent to deciding the equality of the discrete logarithms of T_3, T_3' with bases j, j' respectively. This is infeasible under the decisional Diffie-Hellman assumption over QR_n (Lemma 3.2.7).

Theorem 5.3.6 (Unlinkability). *Under the decisional Diffie-Hellman assumption over QR_n, and the strong collision property of hash function, there exists no probabilistic polynomial-time algorithm that can make the linkability decision for any two arbitrary tuples $(j, T_3), (j', T_3')$ with non-negligible probability.*

5.4. A Scheme for Direct Anonymous Attestation

To provide further flexibility, the propsed group signature scheme can be extended to a DAA-like scheme for trusted computing platforms [13]. To do this, a group member can simply use a value y for which no one knows the discrete log, effectively disabling the open capability. This mirrors the DAA requirement in which opening is forbidden. Furthermore, a variant of the claiming ability can be used for rogue TPM tagging. The current TPM revocation method supported by DAA includes the use of a revocation list that includes any discovered private keys of compromised TPMs. Later, a verifier can identify a rogue TPM by checking the list. For the proposed construction, the private key s should be published on the list. Verifiers can check whether

$$j^s =? T_3 \pmod{n}$$

for all revoked s on the list to identify rogue TPMs (group members).

This section proposes a more efficient way to implement direct anonymous attestation based on the same group membership certificate structure, and shows the new construction is much more efficient than current solution [13] on Sign and Verify protocols, which are the most frequently used protocols in direct anonymous attestation.

5.4.1. More Background on Trusted Computing Platform

TPMs are tamper-resistant cryptographic chips. When a TPM is manufactured, a unique RSA keypair, called the Endorsement Key (EK), is created and stored in the protected area of the TPM. The EK might be generated internally inside a TPM, or imported from an outside key generator. The public part of the EK is authenticated by the manufacturer, while the private part of the EK will never be revealed to the outside. A TPM independently performs cryptographic computations inside itself. Even its manufacturer cannot obtain knowledge of these computations. TPMs are embedded into computing devices by a device manufacturer, and these devices are called trusted computing platforms when coupled with appropriate software. At the heart of trusted computing platform is the assumption that TPMs should independently work as expected, and be "trusted" by remote parties. Essentially, trusted computing platforms are based on trust of TPMs. It is a hardware-assisted technique to enhance computer security.

If the authentication of a TPM is directly based on its EK, all transactions by the same TPM can be linked through the public part of the EK. Furthermore, if the TPM is associated with a user's identity, the user may suffer from loss of privacy. To protect the privacy of a TPM owner, two solutions have been proposed in the TPM specifications.

Privacy in the TPM v1.1 specification is based on a trusted third party, called a Privacy CA. A TPM generates a second RSA keypair called an Attestation Identity Key (AIK). The TPM sends AIK to the Privacy CA, applying for a certificate on the AIK. After the TPM proves its ownership using a valid EK, the Privacy CA issues a certificate for this AIK. Later, the TPM sends the certificate for this AIK to a verifier, and proves it owns this AIK. This way, the TPM hides its identity during the transaction. Obviously, this is not a completely satisfactory solution, since each AIK creation needs the involvement of the Privacy CA, and compromise of the Privacy CA (or a dishonest Privacy CA) can destroy all privacy guarantees.

An alternate solution added in TPM v1.2 is called Direct Anonymous Attestation (DAA), implemented by what we refer to as the BCC scheme, adopting techniques from group signatures: A TPM applies for a credential from an issuer, and later the TPM generate a special signature using this credential. A remote verifier can verify the signature has been constructed from a valid credential without the ability to recover the underlying credential. Different signatures based on the same credential might be linkable or unlinkable depending on a verifier's requirements. If the method implements unlinkable authentication, it is called total anonymity.

Variable anonymity [58] is a conditionally linkable anonymous authentication, in which the signatures signed by the same TPM in a certain time interval are linkable. However, when the signing parameters change, the signatures across the different periods cannot be linked. When the time interval becomes short, the method works like perfectly unlinkable authentication. When the period never expires, this leads to pseudo-anonymity. A verifier can adjust the time interval to detect suspicious attestation. If too many attestation requests come from the same TPM in a period of time, it is likely this TPM has been compromised.

Rogue TPM tagging is about the revocation of the key of a corrupted TPM. When a broken TPM is identified, its secrets (e.g, EK, credential) will be published on the revocation list. A verifier can identify and exclude any rogue TPM on the list, and an issuer refuses to issue new credentials to a TPM with a revoked EK.

The current solution, the BCC scheme, is quite a complex construction with high computing requirements. To expedite authentication, the computation has been distributed between a TPM and the host into which the TPM is embedded. The TPM finishes the computation related to the signature, while the host finishes the computation related to anonymity. The BCC scheme works fine with personal computers with powerful computing capabilities. However, it is still an expensive solution for low end devices.

The following sections presents a new construction that can carry out all the features in DAA and has much more efficient Sign and Verify protocols. However, the Join protocol is less efficient than the BCC scheme. Therefore, the new scheme is more appropriate for low end devices, and settings in which the Join protocol runs rather infrequently.

5.4.2. System Parameter Setting

Besides the system parameter setting for the group signature scheme, additional definitions for Y, l_b and additional range restrictions need to be introduced. The importance of these new parameters will be illustrated in the security proofs shortly (Section 5.4.7).

- l_b: security parameter, greater than 1.
- Y: fixed value. $Y > 2^{\alpha(l_c+l_b)+1}$, and $X > Y + 2^{\alpha(l_c+l_b)} + 2^{\alpha(l_s+l_c)+2}$.
- $(X - 2^{\alpha(l_s+l_c)+1})^2 > X2^{l_c+1} + 2^{\alpha(l_s+l_c)+2}$.

5.4.3. Authentication with Total Anonymity

The idea of the new method to implement authentication with total anonymity is: a TPM picks a random blinding integer b, computes $T_1 = E^b = g^{s^{-1}b} \pmod{n}$, $T_2 = g^b \pmod{n}$. Then the TPM sends (T_1, T_2) to a verifier along with a proof that (T_1, T_2) is constructed from a legitimate keypair.

For a message m, the TPM executes the following steps to complete the Sign protocol:

(i) Generate a random $b \in_R [Y-2^{l_b}, Y+2^{l_b}]$, $t_1 \in_R \pm\{0,1\}^{\alpha(l_s+l_c)}$, $t_2 \in_R \pm\{0,1\}^{\alpha(l_b+l_c)}$, and compute

$$T_1 = E^b \pmod{n},\ T_2 = g^b \pmod{n};\ d_1 = T_1^{t_1} \pmod{n},\ d_2 = g^{t_2} \pmod{n};.$$

(ii) Compute:

$$c = \mathcal{H}_2(g||T_1||T_2||d_1||d_2||m);$$

$$w_1 = t_1 - c(s-X),\ w_2 = t_2 - c(b-Y).$$

(iii) Output (c, w_1, w_2, T_1, T_2).

To verify a signature, the verifier computes

$$c' = \mathcal{H}_2(g||T_1||T_2||T_1^{w_1-cX}T_2^c||g^{w_2-cY}T_2^c||m),$$

and accepts the signature if and only if $c = c'$, $w_1 \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$, and $w_2 \in \pm\{0,1\}^{\alpha(l_b+l_c)+1}$.

5.4.4. Authentication with Variable Anonymity

The TPM derives a generator j of QR_n by hashing the LCID of this signature.

$$j = (\mathcal{H}_1(LCID))^2 \pmod{n}.$$

To implement variable anonymity, the following computations are added to the Sign protocol, i.e., the TPM further computes

$$T_3 = j^s \pmod{n}, \;\; d_3 = j^{t_1} \pmod{n},$$

$$c = \mathcal{H}_2(g||j||T_1||T_2||T_3||d_1||d_2||d_3||m);$$

and outputs $(c, w_1, w_2, T_1, T_2, T_3, m)$. The verifier then computes

$$c' = \mathcal{H}(g||j||T_1||T_2||T_3||T_1^{w_1-cX}T_2^c||g^{w_2-cY}T_2^c||j^{w_1-cX}T_3^c||m).$$

Since j will remain unchanged for a certain time interval, the same TPM will always produce the same T_3 during this interval. The frequency of T_3 will be used by the verifier to identify suspicious authentication, who may refuse to provide further services. Since j changes in different periods of time, this ensures the unlinkability of the same TPM between periods.

5.4.5. Rogue TPM Tagging

As described earlier, TPMs are manufactured to provide tamper-resistance. Otherwise, the basic benefits of trusted computing platforms would become meaningless. However, in extreme circumstances, a TPM may be compromised and its keypair exposed, so a verifier should be able to identify the attestation request from rogue TPMs. To do so, the secrets of a corrupted TPM (e.g., EK, E, and s) should be published on the revocation list. For a keypair (E, s) on the revocation list, a verifier checks

$$T_1^s =? \; T_2 \pmod{n}.$$

If the equation holds, the request comes from a revoked TPM.

5.4.6. Performance Analysis

The computation complexity in the protocol is dominated by the modular squaring and multiplications operations. To estimate the computation cost, it is sufficient to count total modular squarings and multiplications in the protocol. For simplicity, the computation cost are estimated based on techniques for general exponentiation [46]. Let m_1 be the bit length of the binary representation of exponent, and m_2 be the number of 1's in the binary representation. Then the total computation cost can be estimated as m_1 squarings and m_2 multiplications. For example, if $y = g^x \pmod{n}$, and $x \in_R \{0,1\}^{160}$, then the expected number of 1's in x is 80, so the total computation includes 160 squarings and 80 multiplications.

Suppose the group manager sets $\sigma = 1024$, so n is 2048 bits (p, q are 1024 bits). It further chooses $\alpha = 9/8$, $l_c = 160, l_s = 540, l_b = 300$, and sets $X = 2^{792}$ (99 bytes), $Y = 2^{520}$ (65 bytes). This parameter setting conforms to the requirements of the decisional Diffie-Hellman Assumption over subsets of QR_n. It can be observed that most bits of s, b are 0's. The computation with exponent b has 520 squarings and 151 expected multiplications. For authentication with total anonymity, a TPM needs 2352 ($520 \times 3 + 792$) squarings, and 958 ($151 \times 2 + 520/2 + 792/2$) multiplications.

The total exponent bit-length in the BCC scheme is 25844 for authentication with total anonymity. However, due to the computation distribution between a TPM and its host, an efficient exponentiation algorithm has been used in the host part (Algorithm 15.2 in [44]). In their method, the total exponent bit-length for the TPM is around 4088, and 12098 for the host. So the total exponent bit-length is 16186 ($4088 + 12098$), which includes 16186 squarings and 8093 expected multiplications. Assume the cost of squaring is equal to that of multiplication [2], the new scheme is about 7 (24279/3310) times more efficient than the BCC scheme. Even if only the computation inside the TPM is considered, the new scheme is almost 2 (6132/3310) times more efficient than the BCC scheme. For variable anonymity,

[2]Squaring can be at most two times faster than multiplication.

the new scheme needs 5561 modular multiplications, which still can be carried out by the TPM alone.

It should be noticed that distribution of the computation can also be deployed in the new scheme. T_1, T_2, d_2, w_2 can be calculated by the host, and T_3, d_1, d_3, w_1 must be computed inside the TPM. General speaking, this is unnecessary since all the computation can be done by a TPM alone.

Without the distribution of computation, the system design can be greatly simplified. Thus, the new method is more appropriate for mobile devices with low computing capabilities.

5.4.7. Security Properties

In step 5 of the proof about forgery resistance (Theorem 5.3.4), one can obtain a corollary as follows.

Lemma 5.4.1. *Under the strong RSA assumption, it is intractable to forge a keypair* (E, s) *such that* s *lies in the interval* $(0, X - 2^{\alpha(l_s+l_c)+1})$ *or* $(X + 2^{\alpha(l_s+l_c)+1}, (X - 2^{\alpha(l_s+l_c)+1})^2)$*, and* $E^s = g \pmod{n}$.

Proof: In step 5 of the proof for Theorem 5.3.4, if $s \in (0, X - 2^{\alpha(l_s+l_c)+1})$, since all $s_i \in [X - 2^{\alpha(l_s+l_c)+1}, X + 2^{\alpha(l_s+l_c)+1}]$ are prime, then $GCD(s, s_1 s_2 \cdots s_k) = 1$, and one can solve a flexible RSA problem.

If $s \in (X + 2^{\alpha(l_s+l_c)+1}, (X - 2^{\alpha(l_s+l_c)+1})^2)$, due to Lemma 5.3.2, s can not be the product of any $s_i s_j$, $i, j < k$. Thus the proof is as before to solve a flexible RSA problem.

Therefore, under the strong RSA assumption, one can have the lemma as above.

$\square$

The following lemma will be used for the security proof of the new protocol.

Lemma 5.4.2. *Let* n *be an integer. For given values* $u, v \in Z_n^*$ *and* $e, r \in Z_n$ *such that* $e > r$ *and* $v^e = u^r \pmod{n}$*, there is an efficient way to compute the value* (x, y) *such that* $x^y = u \pmod{n}$.

Proof: When $e > r$, there are three cases:

(i) Suppose $GCD(e, r) = 1$. Due to Lemma 3.2.2, one can find a pair (y, e) such that

$$y^e \equiv u \ (mod \ n).$$

(ii) Suppose $GCD(e, r) = r$. Since $e > r$, $e = kr$ for $k > 1$,

$$v^e \equiv v^{kr} \equiv u^r \ (\text{mod } n), \ v^k \equiv u \ (mod \ n).$$

(iii) Suppose $GCD(e, r) = d$ such that $1 < d < r$. Thus one can have $e = kd$ for $k > 1$. Due to Lemma 3.2.3, one can find (y, k) such that

$$y^k \equiv u \, (\text{mod } n).$$

$\square$

Theorem 5.4.3. *Under the strong RSA assumption, the interactive protocol underlying the Sign and Verify protocol is a statistical zero-knowledge proof in honest-verifier mode that the TPM holds a keypair* (E, s) *such that* $E^s \equiv g \ (\text{mod } n)$ *and* s *lies in the correct interval.*

Proof: The proofs of completeness and statistical zero-knowledge property (simulator) follow the same method as the proof for the group signature in this chapter. Here only the existence of the knowledge extractor is outlined.

In the Sign protocol, the TPM proves $T_2 \equiv g^b \ (\text{mod } n)$, and $b \in [Y - 2^{\alpha(l_c + l_b)}, Y + 2^{\alpha(l_c + l_b)}]$. This is a statistical honest-verifier zero-knowledge protocol that is secure under the strong RSA assumption. b can be recovered by a knowledge extractor following the standard method.

It needs to show a knowledge extractor is able to recover the legitimate keypair once it has found two accepting tuples. Let (T_1, T_2, d_1, c, w_1), $(T_1, T_2, d_1, c', w_1')$ be two accepting tuples. Without loss of generality, assume $c > c'$. Then one can have

$$T_1^{w_1 - cX} T_2^c \equiv T_1^{w_1' - c'X} T_2^{c'} \equiv d_1 \ (\text{mod } n).$$

It follows that

$$T_1^{(w_1'-w_1)+(c-c')X} \equiv T_2^{c-c'} \equiv g^{b(c-c')} \pmod{n}. \tag{5}$$

By the system parameter settings, $X > Y + 2^{\alpha(l_c+l_b)} + 2^{\alpha(l_s+l_c)+2}$. Then

$$(c-c')X > (c-c')(Y + 2^{\alpha(l_c+l_b)} + 2^{\alpha(l_s+l_c)+2}).$$

Since $Y + 2^{\alpha(l_c+l_b)} > b$,

$$(c-c')X > (c-c')(b + 2^{\alpha(l_s+l_c)+2}).$$

Since $w_1, w_1' \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$, $w_1' - w_1$ is at least $-2^{\alpha(l_s+l_c)+2}$. Since $c - c'$ is at least 1, finally

$$(w_1' - w_1) + (c-c')X > b(c-c').$$

Due to Lemma 5.4.2, Equation 5 can be solved to obtain a pair (E, s) such $E^s \equiv g \pmod{n}$, $s \leq (w_1' - w_1) + (c-c')X$.

In the parameter settings, $(w_1' - w_1) + (c-c')X < (X - 2^{\alpha(l_s+l_c)+1})^2$. Due to Lemma 5.4.1, s must be a legitimate keypair in the correct interval. Therefore, (E, s) is a valid keypair, which completes the proof.

□

For variable anonymity, $(h, T_3, d_3; T_1, T_2, d_1)$ are used to prove equality of the discrete logarithms of T_3 with base h, and T_2 with base T_1. This is also a statistical honest-verifier zero-knowledge protocol which has been proved secure under the strong RSA assumption.

Theorem 5.4.4 (Anonymity). *Under the decisional Diffie-Hellman assumption over subset of QR_n, the protocol implements anonymous authentication such that it is infeasible to link the transactions by a TPM with different LCID.*

Proof: To decide whether two transactions are linked to a TPM, one needs to decide whether two equations are produced from the same E.

$$T_1,\ T_2 \equiv g^b \equiv T_1^s \pmod{n}$$

$$T_1',\ T_2' \equiv g^{b'} \equiv (T_1')^s \pmod{n}$$

Since T_1, T_1' are random generators of QR_n, By Lemma 3.2.7, it is infeasible to decide whether or not there exist an s such that $T_1^s = T_2$, and $(T_1')^s = T_2'$. The same argument can be applied to variable anonymity, in which case

$$T_3 \equiv j^s \pmod{n},\ T_3' \equiv j'^s \pmod{n}$$

where j, j' are two random generators of QR_n in different periods of time.

□

5.5. Summary

This chapter presented a group signature scheme which is an enhancement of the CM scheme [16] that supports additional features. The new construction supports signature claiming, in which a group member can voluntarily remove the cloak of anonymity from one of their signatures. The proposed scheme also supports the important property of variable linkability, a property which comes from work on anonymous authentication for trusted computing platforms, and is vital for detecting key sharing in an anonymous authentication system. The "variable" part of this can be adjusted to provide a wide range of linkability properties, from completely unlinkable signatures, to signatures linkable within a fixed time period, to completely linkable signatures (giving what is essentially a fixed pseudonym system). In practice, the amount of linkability would be determined by a risk analysis of the application, balancing the goal of protecting a user's privacy against a providers goal of detecting inappropriate uses of keys. As the new scheme supports the full range of linkability options, it provides the best available flexibility to users as well as providers.

The methods to construct a DAA-like scheme for Trusted Computer Platforms has also been introduced, i.e., an anonymous signature scheme without openability. The new DAA scheme has much more efficient Sign and Verify protocols. This new scheme is significantly more efficient than the current solution, making it more appropriate for the devices with lower computing capabilities, and for settings in which the Join protocol is used rather infrequently, which should be the case for most situations.

Finally, the proposed constructions are proved secure under the strong RSA assumption and the Decisional Diffie-Hellman assumption over QR_n.

CHAPTER 6

TRACEABLE SIGNATURE

6.1. Introduction

6.1.1. A General Scenario for Anonymous Authentication

The previous chapter introduced a group signature scheme with variable linkability, and showed how to extend it to a construction for direct anonymous attestation in Trusted Computing Platforms. These constructions work fine in certain specific settings. However, one feature not provided in these schemes is a tracing trapdoor, an important concept introduced in traceable signatures [40]. The combination of traceability and variable anonymity to be particularly important for a more general anonymous authentication system: variable anonymity provides a mechanism to detect key sharing violations, while traceability is an efficient and fair way to reveal all malicious behaviors.

To illustrate the importance of this, recall the anonymous credential sharing problem in the last chapter: a large set of users could share a single subscription to some pay web site. The mechanism of variable anonymity provides a method to detect potential keypair sharing. If during a limited time period an unusually high number of requests using the same credential come in from different IP addresses, then this could be flagged as potentially malicious behavior. The open authority can then reveal the identity of the malicious or compromised group member. A further action would be an investigation of all behaviors by this group member. However, the lack of a tracing trapdoor would make this task difficult: the open authority has to open all signatures to identify all behaviors by this member. As argued in [40], this is either inefficient (a centralized operation by the open authority), or unfair (unnecessarily identifying all innocent group members' signatures). Therefore, it would be better to integrate an explicit tracing trapdoor into an anonymous authentication system. Unfortunately, the protocols in the last chapter do not provide such mechanism. In direct anonymous attestation, the secrets of corrupted TPMs are used to revoke rogue TPMs if these secrets can be somehow extracted out, which might not be always possible. In general settings, an explicit tracing trapdoor is more desirable to overcome the difficulty of extracting a group member's secret. With knowledge of tracing trapdoors for all group members, the group manager can output the tracing trapdoor for a malicious group member to trace agents. Then trace agents reveal all behaviors associated with a trapdoor distributively, and fairly. Also, this tracing trapdoor can be published on the revocation list for verifiers to identify future requests by this corrupted member.

6.1.2. The Results

This chapter presents a construction for traceable signature that supports variable anonymity. The new construction is built up from the well-known ACJT group signature [2]. The traceable signature due to Kiayias et al., which we refer to as the KTY scheme in this paper, is also built up from the ACJT scheme. However, the new construction improves on the KTY scheme in three aspects. First, the same group membership certificate is adopted as in the ACJT scheme. The KTY scheme changes the group certificate in the ACJT scheme to integrate the tracing trapdoor. It can be analyzed this change is unnecessary since tracing trapdoors in fact are already available in the ACJT scheme. Second, the tracing mechanism in the new construction is more efficient than the KTY scheme. The new scheme uses a hash function to create generators while the KTY scheme uses expensive exponentiation computation. Finally, the new scheme supports variable anonymity while the KTY scheme does not. Thus, the proposed scheme is more efficient and flexible than the KTY scheme.

6.2. The Model

This section introduces the model for traceable signature [40], which is a variant of the group signature model (e.g. [2]). Both these two models have protocols for Setup, Join, Sign,Verify, and Open. Traceable Signature has additional protocols for traceability: Reveal, Trace, Claim (Self-trace) and Claim-Verify.

Definition 6.2.1. A traceable signature is a digital signature scheme with four types of participants: Group Manager, Group Members, Open Authorities, and Trace Agents. It consists of the following procedures:

- **Setup**: For a given security parameter σ, the group manager produces system-wide public parameters and a group manager master key for group membership certificate generation.
- **Join**: An interactive protocol between a user and the group manager. The user obtains a group membership certificate to become a group member. The public certificate and the user's identity information are stored by the group manager in a database for future use.
- **Sign**: Using its group membership certificate and private key, a group member creates a group signature for a message. The signature incorporates a linkability class identifier, LCID, which can be negotiated between the signer and the verifier in an interactive setting.
- **Verify**: A signature is verified to make sure it originates from a legitimate group member without the knowledge of which particular one.
- **Open**: Given a valid signature, an open authority discloses the underlying group membership certificate.
- **Reveal**: The group manager outputs the tracing trapdoor associated with a group membership certificate.
- **Trace**: Trace agents check whether a signature is associated with a tracing trapdoor.
- **Claim (Self-trace)**: A group member creates a proof that he created a particular signature.
- **Claim_Verify**: A party verifies the correctness of the claiming proof.

Similar to group signatures, a traceable signature scheme should satisfy the following properties:

- **Correctness**: Any valid signature can be correctly verified by the Verify protocol and a valid claiming proof can be correctly verified.
- **Forgery-Resistance**: A valid group membership certificate can only be created by a user and the group manger through the Join protocol.
- **Anonymity**: It is infeasible to identify the real signer of a signature except by the open authority or if the signature has been claimed.
- **Unlinkability**: It is infeasible to link two different signatures of the same group member if they incorporate different LCIDs.
- **Non-framing**: No one (including the group manager) can sign a message in such a way that it appears to come from another user if it is opened.
- **Traceability**: Given a tracing trapdoor, trace agents can reveal all signatures associated with the trapdoor. A group member can claim (self-trace) his signatures.

6.3. Traceable Signature

The proposed construction is built upon the ACJT group signature scheme. The same system parameters, group certificates, and Join protocol have been adopted in the new scheme. The Sign and Verify protocols have been changed to support traceability and variable anonymity. In the following presentation, the same notation is used as in the original paper to make it easier for readers to see how the proposed method converts the ACJT scheme into a traceable signature scheme.

6.3.1. The System Parameters

The following system parameters are set up when the system is initialized and the group manager key is generated.

- A special RSA modulus $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$, with p, p', q, q' all prime

- Random elements $a, a_0, g \in QR_n$ of order $p'q'$, i.e., these numbers are generators of QR_n
- Security parameters used in protocols: $\epsilon > 1, k, l_p$
- Length parameters $\lambda_1, \lambda_2, \gamma_1, \gamma_2$. $\lambda_1 > \epsilon(\lambda_2 + k) + 2$, $\lambda_2 > 4l_p$, $\gamma_1 > \epsilon(\gamma_2 + k) + 2$, and $\gamma_2 > \lambda_1 + 2$
- Integer ranges $\Lambda =]2^{\lambda_1} - 2^{\lambda_2}, 2^{\lambda_1} + 2^{\lambda_2}[$ and $\Gamma =]2^{\gamma_1} - 2^{\gamma_2}, 2^{\gamma_1} + 2^{\gamma_2}[$
- Three strong collision-resistant hash functions: $\mathcal{H}_1, \mathcal{H}_2 : \{0,1\}^* \to Z_n^*$, and $\mathcal{H}_3 : \{0,1\}^* \to \{0,1\}^k$
- A message to be signed: $m \in \{0,1\}^*$
- The public parameters are (n, a, a_0, g).
- The secret parameters for the group manager are (p', q').

The open authority creates his ElGamal public keypair [27], i.e., private key x and public key y such that $y = g^x \pmod{n}$.

6.3.2. Join Protocol

The same Join protocol is adopted as in the original scheme. A group membership certificate is in the form of $A_i = (a^{x_i} a_0)^{1/e_i} \pmod{n}$ where $x_i \in \Lambda$ is the secret of the group member, and $e_i \in_R \Gamma$ is a random prime number that is known to both the group member and group manager.[1]

In the new scheme, e_i is treated as tracing trapdoor, and kept secret by the group member and group manager. When an open authority reveals A_i for a signature, the group manager sends the corresponding e_i to the trace agents in order to trace all signatures associated with e_i.

[1]Kiayias et al. have showed the range of x_i, e_i can be much smaller without compromising the scheme's security [40]. For simplicity, we still follow the definition in ACJT scheme.

x_i is treated as self-tracing trapdoor, which is used by a group member to claim his signatures. Since x_i is the secret of the group member, only the group member himself has the ability to claim his signatures.

6.3.3. Sign Protocol

In order to sign a message m, a group member does the following:

(i) Derive two generators i and j of QR_n by hashing the LCID of this signature.

$$i = (\mathcal{H}_1(LCID))^2 \pmod{n},\ j = (\mathcal{H}_2(LCID))^2 \pmod{n}.$$

In the random oracle model, with the hash functions modeled by random oracles, each distinct LCID results in i and j being random generators of QR_n with overwhelming probability.

(ii) Generate a random value $w \in_R \{0,1\}^{2l_p}$ and compute:

$$T_1 = A_i y^w \pmod{n},\ T_2 = g^w \pmod{n},\ T_3 = i^{e_i} \pmod{n},\ T_4 = j^{x_i} \pmod{n}$$

(iii) Randomly (uniformly) choose $r_1 \in_R \pm\{0,1\}^{\epsilon(\gamma_2+k)}$, $r_2 \in_R \pm\{0,1\}^{\epsilon(\lambda_2+k)}$, and $r_3 \in_R \pm\{0,1\}^{\epsilon(\lambda_1+2l_p+k+1)}$, and compute

- $d_1 = T_1^{r_1}/(a^{r_2}y^{r_3}) \pmod{n}$, $d_2 = T_2^{r_1}/g^{r_3} \pmod{n}$, $d_3 = i^{r_1} \pmod{n}$, $d_4 = j^{r_2} \pmod{n}$.
- $c = \mathcal{H}_3(g||i||j||y||a_0||a||T_1||T_2||T_3||d_1||d_2||d_3||d_4||m)$;
- $s_1 = r_1 - c(e_i - 2^{\gamma_1})$, $s_2 = r_2 - c(x_i - 2^{\lambda_1})$, $s_3 = r_3 - ce_iw$ (all in Z_n).

(iv) Output the signature tuple $(LCID, c, s_1, s_2, s_3, T_1, T_2, T_3, T_4)$.

6.3.4. Verify Protocol

To verify a signature $(LCID, c, s_1, s_2, s_3, T_1, T_2, T_3, T_4)$, a verifier does the following.

(i) Compute the same generators i and j, and then

$$c' = \mathcal{H}_3(g||i||j||a_0||a||T_1||T_2||T_3||T_4||a_0^c T_1^{s_1-c2^{\gamma_1}}/(a^{s_2-c2^{\lambda_1}}y^{s_3})||$$

$$T_2^{s_1-c2^{\gamma_1}}/g^{s_3}||i^{s_1-c2^{\gamma_1}}T_3^c||j^{s_2-c2^{\lambda_1}}T_4^c||m)$$

(ii) Accept the signature if and only if $c = c'$ and $s_1 \in \pm\{0,1\}^{\epsilon(\gamma_2+k)+1}$, $s_2 \in \pm\{0,1\}^{\epsilon(\lambda_2+k)+1}$, $s_3 \in \pm\{0,1\}^{\epsilon(\lambda_1+2l_p+k+1)+1}$.

6.3.5. Open and Reveal Protocol

For a valid signature, the open authority opens a signature to find its originator by ElGamal decryption:

$$A_i = T_1/T_2^x \pmod{n}.$$

For the non-framing property, the open authority must also issue a proof that it correctly revealed the group member. That is, the open authority proves that the discrete logarithms of y, and $T_1/A_i \pmod{n}$ with respect to g and T_2, respectively, are equal through Protocol 3.3.2.

The opened certificate A_i is submitted to the group manager, and the group manager reveals the corresponding tracing trapdoor e_i to the trace agents.

6.3.6. Trace Protocol

To trace a group member, trace agents use e_i to reveal all the signatures by a group member by checking whether

$$i^{e_i} =? \ T_3 \pmod{n}.$$

To claim a signature, a group member proves its knowledge of discrete logarithm of T_4 with base j through Protocol 3.3.3.

6.4. A Comparison: the ACJT scheme, the KTY scheme and Ours

To better understand the scheme in this chapter, it would be appropriate to illustrate the relationship among the ACJT scheme, the KTY scheme and the new scheme.

As mentioned before, the new scheme directly adopts the same group membership certificate as in the ACJT scheme, which is a pair (A_i, e_i) such that

$$A_i = (a^{x_i} a_0)^{1/e_i} \pmod{n},$$

where x_i, e_i should be in their required ranges, and e_i is a prime number.

In the ACJT scheme, a group member hides (A_i, e_i) by computing

$$T_1 = A_i y^w \pmod{n},\ T_2 = g^w \pmod{n},\ T_3 = g^{e_i} h^w \pmod{n}.$$

Due to the semantic security property of the ElGamal encryption algorithm, it is intractable to decide whether a tuple (T_1, T_2, T_3) is constructed from a specific (A_i, e_i). As argued by Kiayias et al. in [40], the ACJT scheme does not provide certain important features for a realistic anonymous authentication system. Since (A_i, e_i) can not be used to reveal a corrupted group member, the only way to do so is the opening of all signatures in the pool by the open authority. Also, a group member has no way to claim his/her signature if he/she does not record all random number w's in signatures. Obviously, this is not very practical, either.

To overcome these shortcomings, Kiayias et al. introduced a tracing trapdoor into the structure of a group membership certificate. In their scheme,

$$A_i = (b^{x_i'} a^{x_i} a_0)^{e_i^{-1}} \pmod{n},$$

where b is a generator of Z_n^*, and x_i' is a secret shared by a group member and the group manager, which is used as tracing trapdoor. A group member follows the same method as in the ACJT scheme to compute T_1, T_2, T_3, and further computes

$$T_4 = g^{x_i k} \pmod{n}, T_5 = g^k \pmod{n}, T_6 = g^{x_i' k'} \pmod{n}, T_7 = g^{k'} \pmod{n},$$

where k, k' are random elements in Z_n. Obviously T_5, T_7 are two random generators of Z_n^*. Thus, a group member will always produce (T_4, T_5) such as

$$T_4 \equiv T_5^{x_i} \pmod{n},$$

and (T_6, T_7) such as

$$T_6 \equiv T_7^{x_i'} \pmod{n},$$

no matter how k, k' are picked. Since x_i' is the shared secret between a group member and the group manager, if the group manager decides to disclose all behaviors by a group member, x_i' for this group member can be revealed to trace agents to carry out an efficient and fair tracing mechanism by checking

$$T_6 =?\ T_7^{x_i'} \pmod{n}.$$

At the same time, it also becomes trivial for a group member to claim his signature, i.e., he simply issues a zero-knowledge proof for $T_4 = T_5^{x_i}$ for his secret x_i.

The new scheme improves the KTY scheme by identifying that the introduction of x_i' as a tracing trapdoor is unnecessary. Since tracing trapdoor is a shared secret between a group member and the group manager, treating e_i as tracing trapdoor would be the most convenient way. In the ACJT scheme, e_i could be publicly known. However, e_i should be kept secret in the new scheme. This adds a little complexity to the Join protocol, i.e., the Join protocol should be confidential, which is not required in the ACJT scheme.

To carry out traceable mechanism, the same method is used as in the the KTY scheme: create two random random generators i, j, and raise them to i^{e_i}, and j^{x_i}, respectively. To further support variable linkability, the hash function is used for the creation of i, j. Thus, the linkability is controlled by the arguments for the hash function, i.e., LCID (linkability class identifier). The randomness of i, j is achieved by the property of hash function, and the scheme is secure in the random oracle model.

Through these improvements, the new scheme greatly reduces computation cost in the KTY scheme, and further supports variable anonymity, an important malicious behavior detection mechanism in anonymous authentication.

6.5. Security Properties

The proposed scheme uses the same certificate as in the ACJT group signature. The Sign and Verify protocols have been change in the new scheme. The security properties, such as, forgery-resistance, anonymity, non-framing, are unaffected by these changes. In this section, we only discuss the security properties affected by these changes.

The theorem is recalled for the coalition-resistance of group membership certificate in the ACJT scheme. The original proof is quoted with a minor problem being fixed. To avoid many quotation marks which may interrupt and confuse the presentation, explicit quotation marks is not used to show the original work. Note that step 7 in game 1, step 3 and step 7 in game 2, have been re-written. The remaining part is exactly quoted from the original paper.

Theorem 6.5.1 (Coalition-resistance). *Under the strong RSA assumption, a group certificate* $[A_i = (a^{x_i} a_0)^{1/e_i} \pmod{n}, e_i]$ *with* $x \in \Lambda$ *and* $e_i \in \Gamma$ *can be generated only by the group manager provided that the number* K *of certificates the group manager issues is polynomially bounded.*

Proof: Let $\mathcal{M}$ be an attacker that is allowed to adaptively run the JOIN and thereby obtain group certificate $[A_j = (a^{x_j} a_0)^{1/e_i} \pmod{n}]$, $j = 1, \cdots, K$. The task is now to show that if $\mathcal{M}$ outputs a tuple $(\hat{x}; [\hat{A}, \hat{e}])$, with $\hat{x} \in \Lambda, \hat{e} \in \Gamma$, $\hat{A} = (a^{\hat{x}} a_0)^{1/\hat{e}} \pmod{n}$, and $(\hat{x}, \hat{e}) \neq (x_j, e_j)$ for all $1 \leq j \leq K$ with non-negligible probability, then the strong RSA assumption does not hold.

Given a pair (n, z), one can repeatedly play a random one of the following two games with $\mathcal{M}$ and hope to calculate a pair $(u, e) \in Z_n^* \times Z_n$ satisfying $u^e = z \pmod{n}$ from $\mathcal{M}$'s answers. The first game goes as follows:

(i) Select $x_1, \cdots, x_K \in \Lambda$ and $e_1, \cdots, e_K \in \Gamma$.

(ii) Set $a = z^{\prod_{1 \le l \le K} e_l} \pmod{n}$.

(iii) Choose $r \in_R \Lambda$ and set $a_0 = a^r \pmod{n}$.

(iv) For all $1 \le i \le K$, compute $A_i = z^{(x_i + r)\prod_{1 \le l \le K; l \ne i} e_l} \pmod{n}$.

(v) Select $g, h \in_R QR(n)$, $x \in \{1, \cdots, n^2\}$, and set $y = g^x \pmod{n}$.

(vi) Run the JOIN protocol K times with $\mathcal{M}$ on input (n, a, a_0, y, g, h). Assume one is in protocol run i. Receive the commitment C_1 from $\mathcal{M}$. Use the proof of knowledge of a representation of C_1 with respect to g and h to extract $\tilde{x}_i$ and $\tilde{r}_i$ such that $C_1 = g^{\tilde{x}_i} h^{\tilde{r}_i}$ (this involves rewinding of $\mathcal{M}$). Choose α_i and $\beta_i \in]0, 2^{\lambda_2}[$ such that the prepared $x_i = 2^{\lambda_1} + (\alpha_i \tilde{x}_i + \beta_i \mod 2^{\lambda_2})$ and send α_i and β_i to $\mathcal{M}$. Run the rest of the protocol as specified until Step 4. Then send $\mathcal{M}$ the membership certificate $[A_i, e_i]$. After these K registration protocols are done, $\mathcal{M}$ outputs $(\hat{x}; [\hat{A}, \hat{e}])$ with $\hat{x} \in \Lambda, \hat{e} \in \Gamma$, $\hat{A} = (a^{\hat{x}} a_0)^{1/\hat{e}} \pmod{n}$.

(vii) If $gcd(\hat{e}, e_j) \ne 1$ for all $1 \le j \le K$ the output $\perp$ and quit. One can have $\hat{A}^{\hat{e}} = a^{\hat{x}} a_0 = z^{(\hat{x}+r)\prod_{1 \le l \le K} e_l}$. Because $gcd(\hat{e}, e_j) = 1$ for all $1 \le j \le K$, by Lemma 3.2.2, one can find a v, such that $v^{\hat{e}} = z^{\hat{x}+r}$. Due to the choice of x, r, $\hat{e} > \hat{x} + r$, and $1 < gcd(\hat{e}, (\hat{x} + r)) < \hat{e}$. Thus, one can output $u^e = z \pmod{n}$, $1 < e < \hat{e}$ by Lemma 5.4.2.

The previous game is only successful if $\mathcal{M}$ returns a new certificate $[A(\hat{x}), \hat{e}]$, with $gcd(\hat{e}, e_j) = 1$ for all $1 \le j \le K$. Another game can be presented that solves the strong RSA problem in the other case when $gcd(\hat{e}, e_j) \ne 1$ for some $1 \le j \le K$. (Note that $gcd(\hat{e}, e_j) \ne 1$) means $gcd(\hat{e}, e_j) = e_j$ because e_j is prime.)

(i) Select $x_1, \cdots, x_K \in \Lambda$ and $e_1, \cdots, e_K \in \Gamma$.

(ii) Choose $j \in_R \{1, \cdots, K\}$ and set $a = z^{\prod_{1 \le l \le K; l \ne j} e_l} \pmod{n}$.

(iii) Choose a prime number $r \in_R \Lambda$ and set $A_j = a^r \pmod{n}$ and $a_0 = A_j^{e_j} / a^{x_j} \pmod{n}$.

(iv) For all $1 \le i \le K$ $i \ne j$, compute $A_i = z^{(x_i + e_j r - x_j)\prod_{1 \le l \le K; l \ne i,j} e_l} \pmod{n}$.

(v) Select $g, h \in_R QR(n)$, $x \in \{1, \cdots, n^2\}$, and set $y = g^x \pmod{n}$.

(vi) Run the JOIN protocol K times with $\mathcal{M}$ on input (n, a, a_0, y, g, h). Assume one is in protocol run i. Receive the commitment C_1 from $\mathcal{M}$. Use the proof of knowledge of a representation of C_1 with respect to g and h to extract $\tilde{x}_i$ and $\tilde{r}_i$ such that $C_1 = g^{\tilde{x}_i} h^{\tilde{r}_i} \pmod{n}$ (this involves rewinding of $\mathcal{M}$). Choose α_i and $\beta_i \in]0, 2^{\lambda_2}[$ such that the prepared $x_i = 2^{\lambda_1} + (\alpha_i \tilde{x}_i + \beta_i \mod 2^{\lambda_2})$ and send α_i and β_i to $\mathcal{M}$. Run the rest of the protocol as specified until Step 4. Then send $\mathcal{M}$ the membership certificate $[A_i, e_i]$. After these K registration protocols are done, $\mathcal{M}$ outputs $(\hat{x}; [\hat{A}, \hat{e}])$ with $\hat{x} \in \Lambda$, $\hat{e} \in \Gamma$, $\hat{A} = (a^{\hat{x}} a_0)^{1/\hat{e}} \pmod{n}$.

(vii) If $gcd(\hat{e}, e_j) \neq e_j$ output $\perp$ and quit. Otherwise, let $\hat{e} = te_j$ for some t. One can have $\hat{A}^{te_j} = a^{\hat{x}} a_0 \pmod{n}$. Following the definition of a, a_0, $\hat{A}^{te_j} = z^{(\hat{x}+e_j r - x_j)\prod_{1 \leq l \leq K; l \neq j} e_l} \pmod{n}$.

- If $\hat{x} = x_j$, one can obtain $(\hat{A}^{e_j})^t = z^{r \prod_{1 \leq l \leq K} e_l} \pmod{n}$. Since $t < e_j$ for all $1 \leq j \leq K$, r is prime, by Lemma 3.2.2, one can output $u^t = z \pmod{n}$.
- If $\hat{x} \neq x_j$, due to the range restriction for e_j and x, $|\hat{x} - x_j| < e_j$. Then $gcd(\hat{x} + e_j r - x_j, e_j) = 1$. One can obtain $(\hat{A}^t)^{e_j} = z^{(\hat{x}+e_j r - x_j)\prod_{1 \leq l \leq K; l \neq j} e_l} \pmod{n}$. Again by Lemma 3.2.2, one can output $u^{e_j} = z \pmod{n}$.

Consequently, by playing randomly one of the Games 1 or 2 until the result is not $\perp$, an attacker getting access to machine $\mathcal{M}$ can solve the strong RSA problem in expected running-time polynomial in K. Because the latter is assumed to be infeasible, it follows that no one but the group manager can generate group certificates.

□

Next, the security of Sign and Verify protocol is described as the following theorem.

Theorem 6.5.2. *Under the strong RSA assumption, the interactive protocol underlying the group signature scheme is a statistical zero-knowledge (honest-verifier) proof of knowledge of a membership certificate and a corresponding membership secret key.*

Proof: The proof for correctness is straightforward. The proof for zero-knowledge property (simulator) follows the same method as the one for Theorem 5.3.5, demonstrating the existence of a simulator under the Decisional Diffie-Hellman assumption. A knowledge extractor is able to recover the group certificate when it has found two accepting tuples under the same commitment and different challenges from a verifier. Let $(T_1, T_2, T_3, d_1, d_2, d_3, c, s_1, s_2, s_3)$ and $(T_1, T_2, T_3, d_1, d_2, d_3, c', s_1', s_2', s_3')$ be such tuples.

Since $d_4 \equiv j^{s_2 - c2^{\lambda_1}} T_4^c \equiv j'^{s_2 - c'2^{\lambda_1}} T_4^{c'} \pmod{n}$,

$$j^{(s_2' - s_2) + (c - c')2^{\lambda_1}} \equiv T_4^{c - c'} \pmod{n}.$$

By *lemma* 3.2.4, under the strong RSA assumption, $c - c'$ has to divide $(s_2' - s_2) + (c - c')2^{\lambda_1}$. Therefore $\tau_1 = (s_2' - s_2)/(c - c') + 2^{\lambda_1}$.

Since $d_3 \equiv i^{s_1 - c2^{\gamma_1}} T_3^c \equiv i^{s_1' - c'2^{\gamma_1}} T_3^{c'} \pmod{n}$,

$$i^{(s_1' - s_1) + (c - c')2^{\gamma_1}} \equiv T_3^{c - c'} \pmod{n}.$$

Likewise, under the strong RSA assumption, $c - c'$ has to divide $(s_1' - s_1)$, and $\tau_2 = (s_1' - s_1)/(c - c') + 2^{\gamma_1}$.

Since $d_2 \equiv T_2^{s_1 - c2^{\gamma_1}}/g^{s_3} \equiv T_2^{s_1' - c'2^{\gamma_1}}/g^{s_3'} \pmod{n}$,

$$T_2^{(s_1' - s_1) + (c - c')2^{\gamma_1}} \equiv g^{s_3' - s_3} \pmod{n}.$$

Similarly, $\tau_3 = (s_3' - s_3)/((s_1' - s_1) + (c - c')2^{\gamma_1})$.

Since $d_1 \equiv a_0^c T_1^{s_1 - c2^{\gamma_1}}/(a^{s_2 - c2^{\lambda_1}} y^{s_3}) \equiv a_0^{c'} T_1^{s_1 - c'2^{\gamma_1}}/(a^{s_2 - c'2^{\lambda_1}} y^{s_3}) \pmod{n}$,

$$a^{s_2' - s_2 + (c - c')2^{\lambda_1}} a_0^{c - c'} \equiv T_1^{s_1' - s_1 + (c - c')2^{\gamma_1}}/y^{s_3' - s_3} \pmod{n}.$$

Furthermore,

$$a^{(s_2' - s_2)/(c - c') + 2^{\lambda}} a_0 \equiv (T_1/y^{(s_3' - s_3)/((s_1' - s_1) + (c - c')2^{\gamma_1})})^{(s_1' - s_1)/(c - c') + 2^{\gamma_1}} \pmod{n}.$$

Finally, let $A_i = T_1/y^{\tau_3} \pmod{n}$, a valid certificate (A_i, τ_2, τ_1) can be obtained such that $A_i^{\tau_2} = a^{\tau_1} a_0 \pmod{n}$, and τ_1, τ_2 lie in the valid range due to the length restriction on

s_1, s_2, s_3 and c. Therefore a knowledge extractor exist to be able to fully recover a valid group certificate.

□

Unlinkability follows the same argument in the ACJT group signature for T_1, T_2. Since a new T_3, T_4 is defined in the proposed traceable signature, it needs to show this change still keeps the unlinkability property (for different generators i and i'). Similar to the case in the ACJT group signature, the problem of linking two tuples (i, T_3), (i', T_3'), is equivalent to deciding the equality of the discrete logarithms of T_3, T_3' with base i, i' respectively. This is assumed to be infeasible under the decisional Diffie-Hellman assumption over QR_n (Lemma 3.2.7). (j, T_4), (j', T_4') also follows the same argument.

Theorem 6.5.3 (Unlinkability). *Under the decisional Diffie-Hellman assumption over QR_n and with $\mathcal{H}_1$ and $\mathcal{H}_2$ as random oracles, there exists no probabilistic polynomial-time algorithm that can make the linkability decision for any two arbitrary tuples $(i, T_3), (i', T_3')$, or $(j, T_4), (j', T_4')$ with non-negligible probability.*

6.6. Summary

This chapter presents a traceable signature scheme which is an enhancement of the ACJT group signature scheme [2] that supports variable anonymity. The new scheme is a more general solution to anonymous authentication, due to its support of traceability and variable anonymity. Traceability provides an efficient and fair mechanism to reveal and revoke corrupted group members, which is very important to a large, realistic anonymous authentication system. Variable anonymity can be adjusted to provide a wide range of linkability properties, from completely unlinkable signatures, to signatures linkable within a fixed time period, to completely linkable signatures (giving what is essentially a fixed pseudonym system). In practice, the amount of linkability would be determined by a risk analysis of the application, balancing the goal of protecting a user's privacy against a provider's goal of detecting inappropriate uses of keys. As the new scheme supports the full range of linkability options, it provides the best available flexibility to users as well as providers. Finally, the new signature scheme is proved secure under the strong RSA assumption and the Decisional Diffie-Hellman assumption over QR_n.

CHAPTER 7

CONCLUSION

As one of the five security services in the information assurance model, authentication techniques play key roles for information systems. With ever-changing applications, numerous authentication techniques have been proposed to accommodate different scenarios. This dissertation investigates authentication techniques in some emerging areas. Specifically, authentication schemes have been proposed that are well-suited for embedded systems, and privacy-respecting pay web sites.

An efficient public key cryptosystem has been devised, which provides a complete solution to an embedded system, including protocols for authentication, authenticated key exchange, encryption, and revocation. The new construction is especially suitable for the devices with constrained computing capabilities and resources. Compared with other available authentication schemes, such as X.509, identity-based encryption, etc, The new authentication provides unique features such as simplicity, efficiency, forward secrecy, and an efficient re-keying mechanism.

This dissertation extensively discusses anonymous authentication techniques, such as group signature, direct anonymous attestation, and traceable signature. Three anonymous authentication schemes have been proposed which include a group signature scheme with signature claiming and variable linkability, a scheme for direct anonymous attestation in trusted computing platforms with sign and verify protocols nearly seven times more efficient than the current solution, and a state-of-the-art traceable signature scheme with support for variable anonymity. These three schemes greatly advance research in the area of anonymous authentication.

The authentication schemes in this dissertation are based on common mathematical and cryptographic foundations. More specifically, mathematical operations in these schemes are over the quadratic residue group QR_n, where n is a special RSA modulus. The security of these schemes is based on well accepted assumptions related to a group QR_n in cryptography, such as the strong RSA assumption, the computational Diffie-Hellman assumption, and the decisional Diffie-Hellman assumption. The anonymous authentication schemes use an advanced cryptographic tool: zero-knowledge proof of knowledge. In the proposed schemes, zero-knowledge proof of knowledge protocols related to discrete logarithms over QR_n have been intensively used, such as knowledge of a discrete logarithm, knowledge of the equality of two discrete logarithms, and knowledge of a discrete logarithm that lies in an interval.

7.1. Future Research

Even though the schemes in this dissertation have lots of similarities, they are still distinct schemes for different settings. This raises a question: Can we combine these schemes into a unified authentication scheme in which users can be authenticated with different features, either required by verifiers, or chosen by users themselves? For example, a user can adaptively choose a full identification, or a DAA-like authentication in which identity can not be opened, or a complete anonymous authentication in which a group user can not be opened, linked, or traced.

Such a unified authentication scheme could greatly simplify implementation and deployment of an authentication schemes. For instance, this scheme can be implemented on a cryptographic card. Using this card, a user can choose appropriate authentication features under his/her control. This would provide great convenience to users, as well as verifiers. Research on this comprehensive authentication scheme is underway.

BIBLIOGRAPHY

[1] S. Al-Riyami and K. Paterson. Certficateless public key cryptography. In *Advances in Cryptology — Asiacrypt*, pages 452–473, 2003.

[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — Crypto*, pages 255–270, 2000.

[3] G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Financial Cryptography'02*, pages 183–197, 2002.

[4] E. Bach. Discrete logarithm and factoring. Technical Report CSD-84-186, University of California at Berkeley, 1984.

[5] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — Eurocrypto*, pages 480–494, 1997.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient procotols. In *First ACM Conference On computer and Communication Security*, pages 62–73. ACM Press, 1993.

[7] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *A. de Santis, editor, Advances in Cryptology —EUROCRYPT 94, LNCS 950*, pages 92–111. Springer-Verlag, 1995.

[8] D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, pages 48—63, 1998.

[9] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — Crypto'04, LNCS 3152*, pages 41–55, 2004.

[10] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $n^{0.292}$. *IEEE Transactions on Information Theory*, 46(4):1339–1349, 2000.

[11] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto*, pages 213–229, 2001.

[12] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, pages 168–177, 2004.

[13] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM Conference on Computer and Communications Security*, pages 132–145, 2004.

[14] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks (SCN 2004), LNCS 3352*, pages 120–133, 2005.

[15] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN'02, LNCS 2576*, pages 268–289, 2002.

[16] J. Camenisch and M. Michels. A group signature scheme based on an RSA-variants. Technical Report RS-98-27, BRICS, University of Aarhus, Nov. 1998.

[17] J. Camenisch and M. MIchels. A group signature scheme with improved efficiency. In *Advances in Cryptology — ASIACRYPT'98, LNCS 1514*, pages 160–174, 1998.

[18] J. Camenisch and M. Stadler. Efficient group signature schemems for large groups. In *Advances in Cryptology — Crypto'97, LNCS 1294*, pages 410–424, 1997.

[19] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *the 30th ACM Symposium on the Theory of Computing*, pages 209 – 218, 1998.

[20] A. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. In *K. Yyberg, editor, Advances in Cryptology – Eurocrypt'98, LNCS 1403*, pages 561 – 574. Sringer-Verlag, 1998.

[21] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology — CRYPTO'92, LNCS 740*, pages 89–105. Springer-Verlag, 1993.

[22] D. Chaum and E. van Heyst. Group signature. In *Advances in Cryptology — Eurocrypt*, pages 390–407, 1992.

[23] Committee on National Security Systems. National information assurance (IA) glossary – CNSS instruction no. 4009, May 2003.

[24] D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Ueli Maurer, editor, Advances in Cryptology —EUROCRYPT, LNCS 1070*, pages 178–189. Springer-Verlag, 1996.

[25] T. Dieks and C. Allen. RFC2246, the TLS protocols. In *http://www.rfc-editor.org/rfc/rfc2246.txt*, 1998.

[26] W. Diffie and M. E. Hellman. New direction in cryptography. *IEEE Transactions on Information Theory*, 11:644–654, Nov. 1976.

[27] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — Crypto*, pages 10–18, 1984.

[28] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO'86, LNCS 263*, pages 186–194. Springer-Verlag, 1987.

[29] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — Crypto*, pages 16–30, 1997.

[30] E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifable secret sharing and its applications. In *Advances in Cryptology – EUROCRYPTO'98*, pages 32–46, 1998.

[31] H. Ge and S. R. Tate. Efficient authenticated key-exchange for devices with a trusted manager. In *Proceedings of the 3rd International Conference on Information Technology (ITNG) - Embedded Cryptographic Systems track (to appear)*, 2006.

[32] H. Ge and S. R. Tate. A group signature scheme with signature claiming and variable lin ability. In *Proceedings of the 25th IEEE International Performance Computing and Communications Conference, Workshop on Information Assurance (to appear)*, 2006.

[33] R. Gennaro, D. Micciancio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *Proceedings of 5th Conference on Computer & Communications Security*, pages 67 – 72, 1998.

[34] M. Girault. Self-certified public keys. In *Advances in Cryptology — Eurocrypt*, pages 490–497, 1991.

[35] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[36] O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions,. In *Advances in Cryptology — Crypto 84, LNCS 196*, 1984.

[37] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random random functions. *Journal of ACM*, 33(4), 1986.

[38] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of 27th Annual Symposium on Foundation of Computer Science*, pages 291–304, 1985.

[39] ITU-T. ITU-T recommendation X.509 C ISO/IEC 9594-8: Information technology C Open systems interconnection C The directory: Public-key and attribute certificate frameworks. 2001.

[40] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology—Eurocypt, LNCS 3027*, pages 571–589. Springer-Verlag, 2004.

[41] L. Loeb. Information assurance powwow, 2001. http://www-128.ibm.com/developerworks/library/s-confnotes/.

[42] A. Lysyanskaya. Signature schemes and applications to cryptographic protocol design, Sept. 2002. Ph.D. Thesis, M.I.T.

[43] W. V. Maconachy, C. D. Schou, D. Ragsdale, and D. Welch. A model for information assurance: An integrated approach. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, pages 306–310. United States Military Academy, West Point, 2001.

[44] W. Mao. *Modern Cryptography: Theory & Practice*. Prentice Hall PTR, 2004.

[45] J. McCumber. Information systems security: A comprehensive model. In *Proceedings of 14th National Computer Security Conference. National Institue of Standards and Technology, Baltimore.*, 1991.

[46] A. J. Menezes, P. C. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Inc, 1997.

[47] NCSC-TG-017. Guide to understanding identification & authorization in trusted systems. 1991.

[48] M.-W. Online. www.m-w.com/dictionary/privacy, 2006.

[49] H. Petersen and P. Horster. Self-certified keys — concepts and applications. In *Proc. Conf. on Communications and Multimedia Security*, pages 102–116, 1997.

[50] R. L. Rivest and B. Kaliski. RSA problem. In *http://theory.lcs.mit.edu/ rivest/publications.html*, 2003.

[51] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public-key cryptosystems. In *Commnuications of the ACM*, volume 21, pages 120–126, Feb. 1978.

[52] C. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

[53] A. Shamir. On the generation of cryptograpically strong psedorandom sequences. *ACM Transaction on computer systems*, 1, 1983.

[54] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology — Crypto*, pages 47–53, 1984.

[55] V. Shoup. OAEP reconsidered. In *Advances in Cryptology — Crypto01 LNCS 2139*, pages 239–259, 2001.

[56] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.

[57] TCG. http://www.trustedcomputinggroup.org.

[58] TCG. TPM V1.2 Specification Changes: A summary of changes with respect to the v1.1b TPM specification, 2003.

[59] M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

[60] WIKIPEDIA. http://en.wikipedia.org/wiki/Privacy, 2006.