DIRECT ONLINE/OFFLINE DIGITAL SIGNATURE SCHEMES

Ping Yu, M.S.

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2008

APPROVED:

Stephen R. Tate, Major Professor
Ian Parberry, Committee Member
Armin R. Mikler, Committee Member
Ram Dantu, Committee Member
Krishna Kavi, Chair of the Department of Computer Science and Engineering
Costas Tsatsoulis, Dean of the College of Engineering
Sandra L. Terrell, Dean of the Robert B. Toulouse School of Graduate Studies

Yu, Ping. Direct Online/Offline Digital Signature Schemes. Doctor of Philosophy (Computer Science), December 2008, 119 pp., 11 tables, 7 figures, references, 45 titles.

Online/offline signature schemes are useful in many situations, and two such scenarios are considered in this dissertation: bursty server authentication and embedded device authentication. In this dissertation, new techniques for online/offline signing are introduced, those are applied in a variety of ways for creating online/offline signature schemes, and five different online/offline signature schemes that are proved secure under a variety of models and assumptions are proposed. Two of the proposed five schemes have the best offline or best online performance of any currently known technique, and are particularly well-suited for the scenarios that are considered in this dissertation. To determine if the proposed schemes provide the expected practical improvements, a series of experiments were conducted comparing the proposed schemes with each other and with other state-of-the-art schemes in this area, both on a desktop class computer, and under AVR Studio, a simulation platform for an 8-bit processor that is popular for embedded systems. Under AVR Studio, the proposed SGE scheme using a typical key size for the embedded device authentication scenario, can complete the offline phase in about 24 seconds and then produce a signature (the online phase) in 15 milliseconds, which is the best offline performance of any known signature scheme that has been proven secure in the standard model. In the tests on a desktop class computer, the proposed SGS scheme, which has the best online performance and is designed for the bursty server authentication scenario, generated 469,109 signatures per second, and the Schnorr scheme (the next best scheme in terms of online performance) generated only 223,548 signatures. The experimental results demonstrate that the SGE and SGS schemes are the most efficient techniques for embedded device authentication and bursty server authentication, respectively.

ACKNOWLEDGEMENTS

This dissertation would not have been finished without my advisor, Dr. Stephen Tate. I would like to thank him for his continuous encouragement, patience, kindness, and support during my work on the dissertation. I would also like to thank Dr. Ram Dantu, Dr. Armin Mikler and Dr. Ian Parberry for being on my committee and their encouragement.

I also thank my family and friends for their understanding and encouragement throughout these years. It would be impossible for me to finish this dissertation without their tremendous support.

Last but not least, I would like to thank the Department of Computer Science and Engineering and the College of Engineering for their financial support in the past few years.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

Chapters

1. INTRODUCTION 1
   - 1.1 Digital Signature and Online/Offline Signing 1
   - 1.2 Motivation for the Dissertation 4
   - 1.3 Modern Cryptography Basics 7
   - 1.4 Previous Work 11
   - 1.5 Overview of this Dissertation 16

2. PRELIMINARIES 27
   - 2.1 Mathematical Structures 27
   - 2.2 Cryptographic Primitives 31
   - 2.3 Complexity Assumptions 38
   - 2.4 Models of Computation 40

3. SIGNATURE SCHEMES IN THE RANDOM ORACLE MODEL 43
   - 3.1 Design Technique 43
   - 3.2 The RQ Scheme 46
   - 3.3 The RG Scheme 58
   - 3.4 Summary 64

4. SIGNATURE SCHEMES IN THE STANDARD MODEL 66
   - 4.1 Design Technique 66
   - 4.2 Basic Signature Scheme 67
   - 4.3 Signature Scheme for Embedded Device Authentication 73
   - 4.4 Signature Scheme for Bursty Server Authentication 80
   - 4.5 Summary 88

5. EXPERIMENTS 90
5.1 Performance Analysis 91
5.2 Overview of the Experiments 92
5.3 Experimental Results 99
5.4 Summary 109
6. CONCLUSION 110
6.1 Future Research 112
BIBLIOGRAPHY 114

LIST OF TABLES

Page

1.1 Direct Online/Offline Signature Schemes in this Dissertation 22

1.2 Online/Offline Time Complexity 24

1.3 Experiments on a Desktop Computer (times in microseconds on a Pentium Core 2 Duo E6750) 24

1.4 Simulation Experiments on a Embedded Device (times in seconds on AVR Studio) 25

4.1 Comparison of the CL and SQ schemes 73

5.1 Online Performance Comparison of Traditional Signature Schemes and their ST-based Signature Schemes (times in seconds on a Pentium Core 2 Duo E6750) 100

5.2 Offline Performance of Online/Offline Signature Schemes (times in seconds on a Pentium Core 2 Duo E6750) 103

5.3 Online/Offline Performance (times in seconds on AVR Studio) 103

5.4 Online Performance of Online/Offline Signature Schemes (times in microseconds on a Pentium Core 2 Duo E6750) 105

5.5 Bursty Authentication Throughput (online signatures per second) 107

5.6 Verification Performance (times in seconds on a Pentium Core 2 Duo E6750) 108

LIST OF FIGURES

Page

1.1 The Relationship of Security Assumptions....19

5.1 Online Performance Comparison of Traditional Signature Schemes and their ST-based Signature Schemes....101

5.2 Online Performance of the ST-based Signature Schemes....101

5.3 Offline Performance of Direct Online/Offline Signature Schemes....104

5.4 Online Performance of Online/Offline Signature Schemes....105

5.5 Bursty Authentication Throughput....107

5.6 Verification Performance....108

CHAPTER 1

INTRODUCTION

It has been more than thirty years since the digital signature concept was introduced in 1976 [14]. However, research on digital signatures is still active in the cryptographic research community, focusing on either new constructions based on different computational hardness assumptions, or constructions specialized for certain applications in which currently available signature constructions do not perform very well. In this dissertation, signature schemes for applications which demand that a signature be produced very quickly after the message to be signed is known, are considered, which poses a big challenge to the design of a signature scheme. The research results in this area are presented in this dissertation. This chapter reviews the concepts of digital signature, online/offline signing, and previous work in this area, and summarizes the contributions of this dissertation. While this chapter provides an overview of the research, the descriptions are at a high level, and detailed formal definitions appear in Chapter 2 and later in the dissertation.

1.1 Digital Signature and Online/Offline Signing

The digital signature concept is a fundamental primitive in modern cryptography, first proposed by Diffie and Hellman [14]. In such schemes, a signer prepares a keypair which includes a signing key sk and a verification key vk. The signing key is kept secret by the signer while the verification key is public for potential verifiers. For a message m, the signer employs a

signing function Sig to produce a string σ using his signing key as follows:

$$\sigma = Sig(sk, m).$$

This string σ is called the signer's signature on this particular message m.

Later a verifier can use the verification function Ver to check the validity of the signature σ on the message m using the signer's verification key vk as follows:

$$Ver(vk, m, \sigma) \in \{accept, reject\}$$

The concept of online/offline digital signature was first introduced by Even, Goldreich, and Micali in 1989 [17]. An online/offline signature scheme produces a signature in two phases (an offline phase and an online phase). In the offline phase, the signer does most of its work for a signature generation, without knowing the message to be signed. Creation of the signature is completed in the online phase when the message is known, and the computing cost is very low for the online phase so it can be finished quickly. This two-phase signing can be represented as follows.

In the offline phase, the signer typically generates a random input r, and calculates an intermediate value σ' using an offline function $Sig_{offline}$.

$$\sigma' = Sig_{offline}(sk, r).$$

In the online phase, the signer uses an online function Sig_{online} to produce the signature

σ for a message m as follows:

$$\sigma = Sig_{online}(sk, m, \sigma').$$

As an introduction to digital signature schemes and the problems they present in certain settings, the well-known RSA signature scheme is described, which was proposed by Rivest, Shamir and Adleman in 1978, and is still widely used [35]. The following introduction is commonly referred to as the "textbook" version of the RSA scheme since most textbooks introduce RSA this way; however, as it can be seen later, this is not secure and so a modified version is used in practice.

A signer Alice picks two large random prime numbers p and q. She computes $n = pq$, and $\phi(n) = (p-1)(q-1)$, which is called Euler's totient function (n in this form is called an RSA modulus in the literature). She further picks an $e < n$ relatively prime to $\phi(n)$, and calculates d, the multiplicative inverse of e modulo $\phi(n)$. d is Alice's private signing key, and (e, n) is the public verification key. The 4-tuple $(d, p, q, \phi(n))$ is kept secret by Alice. To sign a message, represented by a number $m < n$, Alice calculates

$$\sigma = m^d \mod n,$$

so σ is the signature on m. To verify Alice's signature σ, a verifier Bob does the following computation:

$$m' = \sigma^e \mod n.$$

If $m' = m$, σ is indeed the signature on m produced by Alice. Otherwise, the signature is invalid.

For a typical key size, where n is 1024 bits, computing the signature requires raising a 1024-bit value to a 1024-bit exponent with a 1024-bit modulus, which requires on average 1024 large modular multiplications, so can take considerable time, especially on small devices using relatively weak embedded processors.

1.2 Motivation for the Dissertation

In this dissertation, efficient online/offline digital signature schemes are pursued for two scenarios: bursty server authentication and embedded device authentication. Traditional signature schemes such as the RSA scheme are not efficient for these applications. Furthermore, as it is presented shortly, currently available online/offline techniques overcome many difficulties, but leave room for improvement, and the proposed solutions in this dissertation provide substantial improvements over these techniques.

1.2.1 Scenario 1: Bursty Server Authentication

On the Internet, a server may experience bursty traffic so that at times it needs to authenticate itself simultaneously to a large number of client applications running on remote client machines, while at other times there are many idle CPU cycles. For example, this server could be a stock broker's server and a client wants to confirm that the remote server is not a spoofed server. The standard way of achieving this goal is with a digital signature scheme: the client generates a random nonce, and asks the server to produce a digital signature on this nonce using the server's private key. Then the client can authenticate the server using the server's public key to check if the signature is valid on its random challenge. On a stock broker's server there would be a steady stream of requests during the trading day, but there

might be times — such as immediately after financial updates or news releases — when the number of requests is extraordinarily high, and after hours there is significantly less traffic. Since a traditional signature scheme is time consuming, with a high volume of simultaneous authentication requests the server could be overwhelmed, and not be able to produce signatures in an acceptable amount of time.

1.2.2 Scenario 2: Embedded Device Authentication

Now consider a different scenario in which a mobile device with limited computing capability needs to authenticate itself before accessing certain network resources. A remote server may challenge a device with a random message, asking the device to produce a signature for its message. Since a mobile device has limited computing capability, it might have some difficulty generating the required signature quickly. For example, the authenticating device could be a smart card, with a very weak processor, but which can be loaded with precomputed results of the offline phase from a more powerful device.

1.2.3 Requirements for these Scenarios

To enable solutions for the above scenarios, it is highly desirable that online/offline signing be deployed in a signature scheme to expedite the authentication process: when the system is idle, the majority of computation of the signature can be pre-computed in the offline phase; when a message arrives, only a very simple calculation is needed to complete the signature generation.

However, the criteria is different in terms of what could be the best choice when multiple online/offline constructions are available. For bursty server authentication, we would like the

online computation to be extremely efficient so the authentication throughput could be as large as possible. Therefore online performance is much more important than other factors such as offline signing performance or signature size. Suppose we have two online/offline signature schemes: the first one has better online performance than the second one, but its offline computing takes more time. As long as the offline time is reasonable, we might still prefer the first one due to its better online performance. The reason is that a server is powerful enough to take care of more offline computing overhead during idle time (which is essentially free computation for a server), and the bottleneck of performance is on the throughput for authentication.

In the scenario for embedded device authentication, offline performance can be as significant a concern as online performance. Since authentication for embedded devices happens only occasionally, an online phase that takes 0.1 second would not make a noticeable difference from an online phase that takes 0.01 second. However, many online/offline signature schemes add significant computing overhead in the offline phase (e.g., the Shamir-Tauman method [39]), which might become a problem for embedded devices since the offline computing cost is also vital to these devices. One reason is that these devices have limited computing capabilities, so it is always better to have a scheme with lower total computing cost. Another reason is that most embedded devices are powered by batteries, so more computation implies more power consumption, and shorter life cycle. Therefore, idle time computation is not free as it is for servers, and low offline computation cost is also an important factor for embedded devices.

In summary, Scenario 1 is primarily concerned with online performance, while Scenario 2 must consider more carefully the costs of both offline and online phases.

1.3 Modern Cryptography Basics

Design of a cryptographic construction is a big challenge. After many years of research and practice, standard practice in cryptographic research addresses problems using definitions and analysis with four components, which include definition of a scheme, security properties, computational hardness assumptions, and model of computation.

These notions are further described in the following sections, with informal examples to illustrate the concepts. Formal definitions for the problems studied in this dissertation will be given in Chapter 2.

1.3.1 Definition of a Scheme

The first step of designing a cryptographic construction is to define the functionality for the proposed scheme. Most cryptographic constructions include one algorithm for system parameter generation, and other algorithms to implement the main functionality. For example, the definition of the RSA signature scheme includes an algorithm that produces all system parameters such as $p, q, n, \phi(n), e, d$, a signing algorithm for signature generation, and a verification algorithm for signature verification.

One requirement for a construction is that all algorithms should be carried out as efficiently as possible. The size of operations in a cryptographic scheme is usually controlled by a parameter known as the security parameter, which controls the security and the efficiency of the construction. Implementation is often a balancing act between efficiency (which gets worse as the security parameter increases) and security (which gets better as the security parameter increases). For example, in the RSA scheme, the security parameter is the bit

length of the primes p, q. With larger p, q, we have a stronger but less efficient instantiation of the RSA scheme. A great deal of research effort in cryptography focuses on efficient construction while keeping an appropriate level of security. This is what is pursued in this dissertation.

1.3.2 Security Properties

The second step in designing a cryptographic scheme is to define the desired security properties. For example, common cryptographic problems such as encryption and signatures have well-accepted and well-known security properties, such as security against chosen ciphertext attacks and existential unforgeability.

Security properties are usually defined in terms of an attack game. The game is defined in such a way that the attacker wins if he violates the desired security properties. For example, the definition of secure digital signature was introduced by Goldwasser *et al.* in 1988 under a notion called *existential unforgeability under adaptive chosen message attacks* [24]. Under this notion, the attack game works as follows: an attacker, called the signature forger, is allowed to adaptively choose messages a polynomial number of times, asking the signer to produce the signatures for these messages. No restriction is placed on how a message may be chosen by the signature forger. At the end of the game, if the forger can create a signature which was not produced by the signer, the forger wins the attack game, so the signature scheme is broken. If no polynomial time forger can win the game, the signature scheme is considered to be a secure construction.

This notion of existential unforgeability under adaptive chosen message attacks has become the standard by which digital signature schemes are judged for whether they are strong

enough to be deployed in a real application. In this dissertation, this notion of security is used to prove the security of the proposed schemes.

1.3.3 Computational Hardness Assumptions

A cryptographic scheme is subject to attacks by adversaries, and a secure construction should be able to achieve its security goals under all potential attacks. However, it is a non-trivial task to demonstrate that a construction can satisfy all of its security requirements.

Most security proofs in cryptography follow the so-called "reduction to contradiction" methodology [29]. That is, the proof itself does not enumerate specific attacks. Instead, the proof blindly assumes that there exists a strategy for a probabilistic polynomial time attacker to win the attack game defined in the security properties. Then using this strategy as a "black box", one can devise a method to solve another problem which is believed to be computationally hard. Therefore, the assumed attack strategy is reduced to a method to solve a new problem. If this problem happens to be intractable, meaning that no probabilistic polynomial time algorithm is able to solve it, a contradiction is reached, and the assumption that an attack algorithm exists must not be a valid one. Thus, we can be convinced the proposed scheme is indeed secure.

To use the "reduction to contradiction" method, we need appropriate hard problems. Unfortunately, until progress is made in understanding such fundamental computer science problems as the relation between P and NP, the intractability of such problems must necessarily remain just an assumption. That is, it is assumed that the problem can not be solved in polynomial time by a probabilistic algorithm. Of course, one can not just randomly assume a problem is difficult and devise a construction based on it. An assumption should

be well studied, and several assumptions have become widely accepted in the cryptographic community. For example, factoring has been studied for centuries, and still no one knows how to efficiently factor a number that is the product of two large primes, like an RSA modulus, so this seems like a reasonable assumption. It was initially believed that breaking RSA was equivalent to the difficulty of factoring, but no one has ever been able to prove this (while the availability of an efficient factoring algorithm means you could break RSA, the necessary reduction that breaking RSA means you can factor is still a famous open problem in cryptography). So as a result a new assumption, called the RSA assumption was formulated that does capture the difficulty of breaking RSA. This has subsequently been extended to the strong RSA assumption and the strong RSA subgroup assumption. The proposed constructions in this dissertation rely on these two assumptions, which are introduced in Section 1.5.2.

1.3.4 Model of Computation

A security proof is carried out in a certain context which is called a model in cryptography, which describes what kinds of computational resources the parties in the scheme have access to. The most widely used models in cryptography are the random oracle model and the standard model. The random oracle model was first proposed by Fiat and Shamir [18], and formalized by Bellare and Rogaway [4]. In the random oracle model, a cryptographic hash function is abstracted as a random function that can be accessed by all participants in the protocol, including adversaries. The goal is to establish formal proofs in this model that will carry over to real systems when the random oracle is replaced by a real function. The standard model is also called the real world model, in which the running of the protocol

and the behaviors of the adversary are performed on models of computations that are not augmented with any extra capabilities.

Many digital signature schemes (e.g., Fiat-Shamir [18], Schnorr [38], ElGamal [16], PSS [6]) can be proved secure under adaptive chosen message attack in the random oracle model. However, serious doubt was cast on the random oracle methodology when Canetti *et al.* constructed a scheme that can be proved secure in the random oracle model, while any real implementation will result in an insecure construction [10]. Furthermore, Goldwasser and Kalai recently published a result that casts doubt on the general applicability of the Fiat-Shamir technique [23]. Therefore, security proofs in the random oracle model do not necessarily imply security in the standard model, so a proof of security in the random oracle model can only be treated as a heuristic argument that a scheme is secure.

Due to this fundamental flaw in the random oracle model, it is no longer common to treat a proof in the random oracle model as an air-tight proof in cryptography. As a consequence, providing a proof in the standard model increasingly becomes a standard requirement for any cryptographic construction. However, it is worthwhile to point out that people still use the random oracle model, since it is possible to derive more efficient and simple algorithms this way, even though we should always be somewhat skeptical about the security of such schemes.

1.4 Previous Work

In this section, current solutions which might be deployed in the target scenarios, are reviewed, which include currently available signature schemes, and methods for online/offline signing.

1.4.1 Signature Schemes

Numerous constructions have been proposed in the literature based on different security assumptions. Many schemes are based on the well-known RSA assumption and a variant known as the strong RSA assumption, including PSS [6] and the Cramer-Shoup scheme [12]. Other schemes are based on variants of the discrete logarithm or computational/decisional Diffie-Hellman assumption, including ElGamal signatures [16] and Schnorr signatures [38]. This section introduces signature schemes which are directly related to the research work.

1.4.1.1 The RSA Signature Scheme in the Real World

The "textbook" version of the RSA scheme is introduced in Section 1.1. However, this version of the RSA scheme is not secure with respect to the existentially unforgeability criteria. For example, suppose that we have two signatures such as

$$\sigma_1^e = m_1 \mod n, \quad \sigma_2^e = m_2 \mod n.$$

It is easy to compute a pair (σ_3, m_3) such that

$$\sigma_3 = \sigma_1 \times \sigma_2 \mod n, \quad m_3 = m_1 \times m_2 \mod n, \text{ and } \sigma_3^e = m_3 \mod n.$$

This shows (σ_3, m_3) is also a valid signature. One may argue that m_3 may not be a valid message, but in general we do not assume a message should be in certain format. A message is simply series of bytes.

In practice, the message to be signed is pre-processed in a certain way, then the RSA

function is applied to the processed message to obtain the final signature. Many techniques have been proposed to do this pre-processing, such as PSS and PKCS#1 [6].

1.4.1.2 The Schnorr Signature Scheme

The first signature scheme suitable for devices with limited computing capabilities was devised by Schnorr [38]. Schnorr constructed a three-round identification scheme over a small prime-order subgroup of Z_p^*, which was then converted into a signature scheme using the Fiat-Shamir heuristic [18]. This scheme will be described in Chapter 5.

The Schnorr scheme is an efficient direct online/offline signature construction. However it can only be proved secure in the random oracle model.

1.4.1.3 Signature Schemes Secure in the Standard Model

Even though the first signature scheme was introduced in 1978, the first practical and provable signature scheme in the standard model was invented much later. In 2000 Cramer and Shoup [12] proposed the first practical digital signature scheme secure under adaptive chosen message attacks under the strong RSA assumption in the standard model. Before this construction, available schemes secure under adaptive chosen message attacks in the standard model were not practical for real applications [24, 11, 15]. In this dissertation, Cramer and Shoup's scheme is referred as the CS scheme.

Based on the ideas of Cramer and Shoup, Camenisch and Lysyanskaya [9], Zhu [44, 45], and Fischlin [19] all proposed schemes with similar structure based on the strong RSA assumption. In 2005, Groth extended these results to work over a small subgroup of Z_n^*, improving the efficiency of signature generation [25].

Unfortunately, none of these schemes can operate directly in an online/offline manner.

1.4.2 Methods for Online/Offline Signing

This section reviews two generic methods which can convert any signature scheme into an online/offline construction: the Even-Goldreich-Micali method and the Shamir-Tauman method.

1.4.2.1 The Even-Goldreich-Micali Method

In 1989, Even *et al.* proposed a generic method to convert any signature scheme into an online/offline construction [17]. Their method is based on a special type of digital signature called a one-time signature scheme which was proposed by Lamport [28] and Rabin [34].

In a one-time signature scheme, a signing/verification keypair can only be used once when signing a message, which means that for each new message to be signed, the signer needs to create a new signing/verification keypair. That is why this type of construction is called "one-time." A one-time signature scheme can be implemented using a traditional hash function (e.g., SHA1 [33]), or a symmetric encryption function (e.g., DES [32]). Therefore, one-time signature construction generally is very fast.

The idea of the Even-Goldreich-Micali method is as follows: during the offline phase, a signer prepares as many one-time keypairs (sk, vk) as needed for its one-time signature scheme, and uses a traditional signature scheme to sign these one-time verification keys. In the online phase, when a message is known, the signer picks a one-time signing key to sign the message. Therefore the final signature includes the signature for the one-time verification key in the traditional signature scheme and the signature for the message in the

one-time signature scheme. After creating a signature, this one-time keypair (sk, vk) should be destroyed since it can not be re-used.

One thing making the Even-Goldreich-Micali method impractical is its large signature size. Even *et al.* presented four concrete constructions whose signature lengths vary from 4208 bytes to 31592 bytes, with the longer signature providing stronger security. This long of a signature is normally undesirable for most applications. For comparison, an RSA signature is only 32 or 64 bytes for different parameter choices. The long signature size in the Even-Goldreich-Micali method is mainly due to the one-time signature scheme, which is inherent in the design of the one-time signature construction.

Recently, some results have been published on reducing signature size for a one-time signature [31], although the proposed scheme still generates signature that are several thousand bytes long, which is still not practical enough for many applications.

1.4.2.2 The Shamir-Tauman Method

In 2001, Shamir and Tauman proposed another generic method to achieve online/offline signing, which is quite efficient [39]. Their method is based on a new type of hash function called a trapdoor hash function, which was proposed by Krawczyk and Rabin [27], and allows the use of a "hash-sign-switch" paradigm.

A trapdoor hash function uses a keypair with a public "hash key" hk and a private "trapdoor key" tk, and the hash for message m is produced by the function $h(hk, m, r)$, where r is a supplemental random input. For a secure trapdoor hash function, given hk, m', r', and alternative message m, it is infeasible to find an r such that $h(hk, m, r) = h(hk, m', r')$; however, given tk it is easy to find such an r.

The idea of Shamir and Tauman's "hash-sign-switch" paradigm is to first use any signature algorithm to sign the hash value $h(hk, m', r')$ for a random message m', and then when the real message m is known (in the online phase) the signer simply computes the r so that the hash remains the same and the precomputed signature is valid. The random value r is supplied as part of the signature. This is not suitable for all applications since you need to maintain an extra keypair, and the signature length is increased (due to the inclusion of r).

The Shamir-Tauman method will be introduced with formal definitions in Chapter 2.

1.5 Overview of this Dissertation

So far, current techniques which may provide solutions for the target scenarios have been reviewed. Unfortunately, none of these can fully meet the specified requirements. The goal of this research is to find new techniques to devise online/offline signature schemes which can provide good solutions to the sample scenarios. In this dissertation, new techniques which produce direct online/offline signature schemes with highly efficient online operations, are introduced. These new signature schemes operate under different models and assumptions, so a person using these signature schemes can decide on how they want to balance security assurance versus efficiency in practice.

1.5.1 Direct Online/Offline Signing

The Shamir-Tauman technique for online/offline signing is more appropriate in situations where a signature scheme is already in place, and we only want to "embed" an online/offline signing mechanism into the current system without discarding anything. The downside of this method is that it increases the complexity of the current system by introducing

additional cryptographic settings, and also increases overall computing overhead as well as the signature size.

In many scenarios, including the target scenarios, it would be more desirable for an online/offline signing to be directly implemented in a signature scheme from the beginning, so the whole system could be carried out more simply and efficiently. In fact, many signature schemes naturally have an online/offline mechanism with only minor effort (e.g., The ElGamal Scheme [16] and the Schnorr scheme [38]). But the problem for these direct online/offline signature schemes is that they can only be proved secure in the random oracle model, which is less than ideal, as described in Section 1.3.4.

In Chapter 3, the core design technique for direct online/offline signing is introduced. Based on this new technique, additional online/offline signature schemes that are secure in the random oracle model as well as in the standard model are devised. This way, some of the shortcomings of the Shamir-Tauman method are overcame.

1.5.2 Security Assumptions and Efficiency

As described in Section 1.3.3, the security of cryptographic constructions is based on certain assumptions. The constructions in this dissertation are based on the strong RSA assumption and the related subgroup variant. The strong RSA assumption is a well-accepted cryptographic assumption which was first proposed by Baric and Pfitzmann [3] and Fujisaki and Okamoto [20].

The strong RSA assumption is defined over a mathematical structure, called the modular group Z_n^* (defined in Chapter 2). Informally, the strong RSA assumption states that it is impossible to solve the problem of taking a random number u in the modular group Z_n^* for

n being an RSA modulus as in the RSA signature scheme, and finding a pair (v, e) such that $e > 1$ and $v^e \equiv u \mod n$. This problem is called the flexible RSA problem. For example, suppose that $n = 35$, and when given a random number 27, one is asked to find a pair (v, e) such that $v^e = 27 \mod 35$. This assumption is closely related to the standard RSA assumption in which e is a given number. That is, the RSA problem is to take a pair (e, u) and find a v such that $v^e = u \mod n$. Using the same example, the RSA problem could specify $e = 5$ and ask for a v such that $v^5 = 27 \mod 35$. Obviously, the RSA assumption puts more restrictions on the problem than the strong RSA assumption does. In cryptography, an assumption is stronger if the underlying problem is potentially easier to solve. For example, solving the Flexible RSA problem certainly is no harder than solving the RSA problem, so the strong RSA assumption is a stronger assumption than the RSA assumption (which is also reflected in the naming of the assumptions).

The efficiency of a cryptographic construction is significantly affected by its underlying mathematical structure. Certain operations on Z_n^* (e.g., exponentiation) are computing intensive, resulting in slow signature generation in many schemes. To improve computational efficiency, Groth recently investigated cryptography over a small subgroup of Z_n^*, and introduced a variant of the strong RSA assumption, called the strong RSA subgroup assumption over the small subgroup of Z_n^* [25]. By using a much smaller group, the computation cost can be significantly reduced. Thus Groth's construction improves the efficiency of the cryptographic construction while requiring a stronger assumption. All of these assumptions are called the RSA-type assumptions.

The new schemes presented in this dissertation are based on the strong RSA assumption and the strong RSA subgroup assumption. A question remains: how to determine the

security of the schemes? Or, more specifically, how to choose the security parameter k for the constructions? These questions in turn rely on another assumption called the factorization assumption, which states that it is infeasible to factor an RSA modulus. Obviously, if n can be factorized efficiently, all underlying problems in the RSA-type assumptions can be solved, and these assumptions will not hold on any longer. Therefore, the factorization assumption is weaker than the RSA-type assumptions.

Figure 1.1 shows the relationship of these assumptions, '$<$' means the assumption on the left side is weaker than the one on the right. If a cryptographic construction is based on a stronger assumption, then theoretically there is less assurance of security (although less assurance does not mean that it is easier to break). Conversely, if a construction is devised using a weaker assumption, the construction could be stronger.

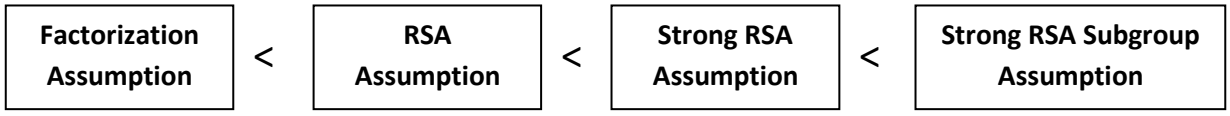

Figure 1.1: The Relationship of Security Assumptions

Even though the RSA-type assumptions are stronger than the factorization assumption, all known methods for solving the underlying problems for these RSA-type assumptions require factoring n first. Therefore, Selection of the security parameter is based on the best known algorithms for factoring when setting the security of schemes which are based on the RSA-type assumptions. Selecting the security parameter for schemes that work over small subgroups is more complicated, and the recommendations of Groth are followed to determine appropriate values for the security parameter.

All current integer factorization algorithms (e.g., Quadratic Sieve Algorithm, Number

Field Sieve Algorithm, etc [30]) are super-polynomial in k for k being the bit length of the two prime factors of n. For example, when $k = 512$, the bit length of p, q is set to 512, so the bit length of n is 1024. Currently, $k = 512$ is considered to be secure for most applications. The largest number of this form ever factored is RSA-200 which is a 663-bit RSA modulus, and it took about 3 months on a cluster of 80 2.2 GHz Opterons [36].

1.5.3 Contributions of this Dissertation

In this dissertation, new techniques for online/offline signing are introduced, those are applied in a variety of ways for creating online/offline signature schemes, and two sets of direct online/offline signature schemes that are proved secure under a variety of models and assumptions are proposed. The first set of constructions can be proved secure in the random oracle model, while the second set of constructions can be proved secure in the standard model. These schemes are called the RQ, RG, SQ, SGE, and SGS schemes. In this dissertation, these schemes are named in following notation: The first letter is either R, or S, which means the scheme can be proved secure either in the $r$andom oracle model, or in the $s$tandard model. The second letter is either Q, or G, which means the scheme is over either QR_n, or the small subgroup G of Z_n^*. An additional letter E or S indicates the scheme is specialized for the $e$mbedded device authentication, or the $s$erver authentication.

Chapter 3 introduces the core design technique used in this dissertation, the two-exponent version of the flexible RSA problem, which provides the flexibility of performing some computations before the message to be signed is known. Subsequently, the first direct online/offline signature scheme using this design technique, the RQ scheme, is devised. This work has been

published in the paper [42]. Even though the RQ scheme is not a fully optimized construction, it builds the foundation for subsequent improvements. Using computation over the small subgroup of Z_n^*, a simple and efficient signature construction, called the RG scheme, is further devised. However, both the RQ and RG schemes can only provide security in the random oracle model.

In Chapter 4, the underlying reason for the random oracle being necessary in the proofs for the RQ and RG schemes is analyzed, and a new design technique is introduced when using the two-exponent version of the flexible RSA problem. Then a direct online/offline signature scheme in the standard model is devised. This construction is called the SQ scheme. Based on the SQ scheme, first computation over the small subgroup of Z_n^* is used to reduce overall computation overhead. Then, a type of function called a division intractable hash function, is used to further reduce the offline cost, and the most efficient construction in the offline phase is obtained. This construction is called the SGE scheme, which targets the scenario of embedded device authentication. The SQ and SGE schemes are published in the paper [43]. For the last technique, Shamir and Tauman's trapdoor hash function is adapted into the direct online/offline method, and the most efficient construction in the online phase is obtained. This construction is called the SGS scheme, which targets the scenario of bursty server authentication.

Table 1.1 summaries all of the new online/offline signature schemes. In the table, "ROM" means the random oracle model while "SM" means the standard model. "SRSA" means the strong RSA assumption while "SRSA-S" means the strong RSA subgroup assumption.

The time complexity for the offline phase is largely measured by the number of modular multiplications in these schemes, which is determined by the security parameter k, and the

Schemes	Secure Model	Assumption	Application Area
RQ	ROM	SRSA	-
RG	ROM	SRSA-S	Generic Scenario
SQ	SM	SRSA	Generic Scenario
SGE	SM	SRSA-S	Embedded Device
SGS	SM	SRSA-S	Server

Table 1.1: Direct Online/Offline Signature Schemes in this Dissertation

cost of generation of a prime number if needed for the scheme. Specifically, the RQ scheme needs about $2k$ modular multiplications to complete offline signing, while the SQ scheme requires about $2k$ modular multiplications and the generation of a 162-bit prime number. The use of a small subgroup in the RG, SGE and SGS schemes results in a significant decrease in the number of modular multiplications required, with these schemes requiring roughly $\frac{2}{5}k$ modular multiplications. The SGE scheme needs additional computation for the generation of an 162-bit prime number, and the SGS scheme needs additional computation for the generation of an 88-bit prime number. As a concrete example, for $k = 512$, n is 1024 bits long. The offline cost is about 1022 modular multiplications (plus additional cost if needed) for the schemes based on the strong RSA assumption, while the cost for the schemes based on the strong RSA subgroup assumption, is reduced to about 200 modular multiplications (plus additional cost if needed).

As for the online phase, all these schemes only need a constant number of relatively simple arithmetic operations, such as addition, multiplication, or modular reduction. However, each operation has different computational cost. Unlike in the offline phase where the security parameter k determines the number of modular multiplications in the schemes, in the online phase k mainly affects the bit length of the operators of a simpler arithmetic operation. For

instance, modular addition is much less expensive than multiplication. Therefore, the online cost is mainly affected by number, type, and the length of the operators in these operations. Then the effort of reducing the online cost focuses on devising efficient online procedure that use fewer operations as well as lower cost operations. The online cost for each scheme is listed when n is a 1024-bit RSA modulus.

- RQ: one 1022-bit number by 1024-bit number modular multiplication with a 1022-bit modulus;
- RG: one 200-bit number by 1024-bit number modular multiplication with a 260-bit modulus, and one modular addition with a 260-bit modulus;
- SQ: one 160-bit number by 1022-bit number multiplication and one addition;
- SGE: three 200-bit number by 60-bit number multiplications, and three additions;
- SGS: one addition, and one modular reduction of a 420-bit number by a 260-bit modulus.

All these schemes can complete signing and verification in polynomial time with respect to k. Table 1.2 shows the time complexity of these schemes, where $m(n_1, n_2)$ is the time needed for one n_1-bit number by n_2-bit number multiplication, $add()$ is the time needed for one addition, $madd(n)$ is the time needed for a modular addition with n-bit modulus, $mm(n)$ is the time for an n-bit modular multiplication, $pg(n)$ is the time needed to generate a n-bit prime number, $inv()$ is the time needed to compute a multiplicative inverse of a number, and $mr(n_1, n_2)$ is the time needed for one n_1-bit number by n_2-bit number modular reduction.

Schemes	Offline Time Complexity	Online Time Complexity
RQ	$2k * mm(2k)$	$mm(2k)$
RG	$\frac{2}{5}k * mm(2k)$	$mm(\frac{2}{5}k + 60) + madd(\frac{2}{5}k + 60)$
SQ	$2k * mm(2k) + pg(162)$	$m(160, 2k) + add()$
SGE	$\frac{2}{5}k * mm(2k)$	$3 * m(60, \frac{2}{5}k) + 3 * add()$
SGS	$\frac{2}{5}k * mm(2k) + pg(88)$	$add() + mr(\frac{2}{5}k + 220, \frac{2}{5}k + 60)$

Table 1.2: Online/Offline Time Complexity

To see the real effect of the proposed schemes, especially performance in the online phase which is not obvious from theoretical analysis, a series of experiments comparing the proposed schemes with each other and with other state-of-the-art schemes in this area, have been conducted, both on a desktop computer environment, and in a simulation environment for embedded devices. Table 1.3 shows one snapshot of the experimental results on a desktop class computer for n being 1024 bits. The RSA-ST scheme is an online/offline RSA construction based on the Shamir-Tauman method.

	RSA	RSA-ST	Schnorr	SGE	SGS
Offline	N/A	79200	7200	9600	10400
Online	40000	4.5	4.5	4.7	2.1

Table 1.3: Experiments on a Desktop Computer (times in microseconds on a Pentium Core 2 Duo E6750)

These results show that the SGS scheme beats all other schemes in terms of online performance. To understand what this improvement means for the bursty sever authentication scenario, we can calculate how many authentications can be completed in one second. The numbers of signatures produced per second by SGS, SGE, Schnorr, RSA-ST, and RSA are 469109, 214348, 223548, 221273, and 25, respectively. Thus, SGS scheme generates 2.19 times as many as the SGE scheme, 2.098 times as many as the Schnorr scheme, and 2.12

times as many as the RSA-ST scheme, and 18764.36 times as many as the (traditional, non-online/offline) RSA scheme. Thus we can see experimentally that the improvement by the SGS scheme is quite significant.

Table 1.4 shows the simulation results on AVR Studio, which simulates an 8-bit processor that is popular in embedded devices. It can be seen the SGE scheme is really suitable for embedded devices with about 24.1 seconds for the offline phase and 14.9 milliseconds for the online phase. The RSA-ST scheme needs 212.2 seconds in the offline phase, which cause more rapid battery drainage (8 times the cost of SGE), and the RSA scheme takes about 106 seconds to finish the online phase, which is unacceptable for a embedded device to authenticate to a network. Thus, the SGE scheme is far more suitable for the embedded device authentication scenario. Notice that for the offline phase the Schnorr scheme runs faster than the SGE scheme, taking 21.7 seconds to finish the offline computation, about 10 percent faster than the SGE scheme. However, this advantage is small considering that the Schnorr scheme can only be proved secure in the random oracle model, while the SGE scheme has a proof in the standard model.

	RSA	RSA-ST	Schnorr	SGE	SGS
Offline	N/A	212.24	21.69	24.11	26.69
Online	105.99	0.0138	0.0191	0.0149	0.0077

Table 1.4: Simulation Experiments on a Embedded Device (times in seconds on AVR Studio)

In summary, a set of new online/offline signature schemes that operate under a variety of models and cryptographic assumptions, including two that are optimized for specific target scenarios, have been successfully devised in this dissertation. The schemes that are devised in this dissertation are currently the best known constructions for online/offline signature

schemes.

1.5.4 Scope of this Dissertation

The rest of the dissertation is organized as follows. Chapter 2 reviews some basic concepts in abstract algebra, formal definitions for some cryptographic primitives, and security proof techniques in cryptography. Chapter 3 presents the core design technique and the two online/offline digital signature schemes that are secure in the random oracle model. Chapter 4 introduces a new design technique and the online/offline digital signature schemes that are secure in the standard model, which include a basic scheme, one construction for embedded device authentication, and one for bursty server authentication. Chapter 5 presents the experimental results. The research results are summarized and discussed, and conclusions and future research for the dissertation are presented in Chapter 6.

CHAPTER 2

PRELIMINARIES

This chapter reviews the important concepts, mathematical structures and techniques in abstract algebra, number theory and cryptography, which have been used throughout this dissertation.

2.1 Mathematical Structures

The concept of groups in mathematics plays a key role in this dissertation. First some basic definitions from abstract algebra, focusing on concepts which are used frequently in this dissertation, are reviewed. Then some special group constructions which have been used in the proposed signature schemes, are introduced. For more information about these concepts and definitions, consult any good abstract algebra book such as Shoup's introductory book that takes a computational approach [41].

2.1.1 Definition of Group

Definition 2.1.1 (Group [30]) *A group $(G, *)$ consists of a set G with a binary operation $*$ (called the group operation) on G such that*

- *The group operation is* associative. *That is, $a * (b * c) = (a * b) * c$, for all $a, b, c \in G$,*
- *There is an element $e \in G$, called* identity element*, such that for all $a \in G$, $a * e = a = e * a$,*

- *For each $a \in G$, there exists an element $a^{-1} \in G$ such that $a * a^{-1} = e = a^{-1} * a$. a^{-1} is called the* inverse *of a.*

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

Definition 2.1.2 (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 of G.*

Definition 2.1.3 (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 $i \geq 0$ with $a = g^i$. Such an element g is called a generator of G.*

Definition 2.1.4 (Order of an Element) *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.*

2.1.2 Modular Groups

For a positive integer n, the modulo operation on n maps all integers into the set $\{0, \ldots, n-1\}$. For two integers a, b, if $a = b + kn$ for an integer k, we write $a \equiv b \mod n$. Modular arithmetic exhibits the following properties:

- $(a \mod n + b \mod n) \mod n = (a + b) \mod n$
- $(a \mod n - b \mod n) \mod n = (a - b) \mod n$
- $(a \mod n \times b \mod n) \mod n = (a \times b) \mod n$

A finite group can be formed by modular arithmetic, and many cryptographic schemes use computation over modular groups, such as Z_n^* and QR_n which are defined as follows.

Definition 2.1.5 (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 the multiplicative group Z_n^*.*

Definition 2.1.6 (Quadratic Residue Group QR_n) *Let Z_n^* be the multiplicative group modulo n. An element $x \in Z_n^*$ is called a* quadratic residue *if and only if there exists an $a \in Z_n^*$ such that $a^2 \equiv x \mod n$. The set of all quadratic residues of Z_n^* forms a cyclic subgroup of Z_n^*, which is denoted by QR_n.*

2.1.3 Groups Used in the Dissertation

One of the first public key cryptographic systems published was the RSA scheme [35], which uses computation over modular group Z_n^*, where $n = pq$, p, q are both prime, and n should be constructed in a way such that factorization of n is infeasible. This type of n is called an RSA modulus. Many different ways exist to construct n such that the resulting modular groups exhibit different properties which can be used in cryptographic constructions.

In this dissertation, two special subgroups of Z_n^* for n being an RSA modulus are used: one is QR_n, another one is G. Before the subgroup G of Z_n^* is introduced, the definition of a special RSA modulus is reviewed.

Definition 2.1.7 (Special RSA Modulus) *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 also called safe prime numbers, and n is called a safe RSA modulus.*

When n is a special RSA modulus, the order of QR_n is exactly one fourth of the order of Z_n^*, i.e., $|QR_n| = \frac{1}{4}|Z_n^*|$. When constructed with a special RSA modulus, QR_n also has a special property described by Camenish and Lysyanskaya which will be used in some later proofs (Lemma B.3 in [9]).

Property 2.1.1 *If n is a special RSA modulus, with p, q, p', and q' as in Definition 2.1.7 above, then $|QR_n| = p'q'$ and $(p'-1)(q'-1)$ elements of QR_n are generators of QR_n.*

Two of the proposed schemes (the RQ and SQ schemes) use the subgroup QR_n of Z_n^*. While QR_n is an useful subgroup of Z_n^* for cryptographic constructions, its size is comparable to the size of Z_n^*, which is fairly large for certain applications where the complexity grows with the size of the group being used.

By using a different form of modulus n, there are smaller subgroups, which has been shown to be useful in recent work by Groth, who investigated cryptography over a small subgroup of Z_n^* [25]. Three of the proposed schemes (the RG, SGE, and SGS schemes) use this special kind of small group of Z_n^*. The definition is presented as introduced by Groth here.

Definition 2.1.8 (Small Subgroup G of Z_n^*) *Let $n = pq$ such that $p = 2p'r_p + 1$ and $q = 2q'r_q + 1$, where p, p', q, q' are all prime. There is a unique cyclic subgroup G of Z_n^* of order $p'q'$. For the purpose of efficient cryptographic construction, the order of G, i.e., $p'q'$, is chosen small. Let g be a random generator of G, and (n, g) is called an RSA subgroup pair.*

2.2 Cryptographic Primitives

This section reviews definitions for some cryptographic primitives such as digital signatures and hash functions. First, some commonly used terminology in theory of computation [22] are reviewed.

2.2.1 Terminology in Theory of Computation

Definition 2.2.1 (Probabilistic Polynomial Time [22]) *A probabilistic Turing machine is a standard Turing machine that is augmented with an extra read-only tape that provides random bits. A probabilistic polynomial time algorithm is a probabilistic Turing machine that always halts in a number of steps that is bounded by a polynomial in the size of the input.*

When defining the security of a cryptographic scheme, we can never guarantee that a scheme cannot be broken — in any public-key scheme it is possible, although highly unlikely, for an attacker simply to randomly select a valid signature. In order to refer to probabilities like this that are extremely small, the concept of a "negligible function" is used.

Definition 2.2.2 (Negligible Function [22]) *A function $v(k)$ is negligible if, for any positive polynomial $p(\cdot)$, there exists an N such that for all $k > N$,*

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

Success probability of attack games is used to represent the security of the scheme, which is affected by the security parameter k of this scheme.

Definition 2.2.3 (Success Probability of Attack Games) *$A(x)$ is an algorithm (possibly given with multiple steps) and $P(x)$ is a predicate that also might have multiple parts. Success probability of attack games is represented as $Pr[A(x), s.t., P(x)]$, which is the probability that condition $P(x)$ holds after running $A(x)$, where the probability is taken over coin tosses made by $A(x)$.*

Computational indistinguishability is an important concept in cryptography. Many security proofs for cryptographic constructions are based on a simulation paradigm, in which a simulator sets up a context, and plays the attack game with the attacker. The primary requirement for this simulation is that it should be infeasible for the attacker to determine whether he is in a simulation context rather than in the real context of the scheme. That is, the simulation context should be computational indistinguishable from the real context. Computational indistinguishability is sometimes called polynomial time indistinguishability, and is defined as follows.

Definition 2.2.4 (Polynomial Time Indistinguishability [29]) *Let $E = \{e_1, e_2, \cdots\}$, $E' = \{e'_1, e'_2, \cdots\}$ be two ensembles in which e_i, e'_i are random variables in a finite sample space S. Denote $k = log_2|S|$, where $|S|$ is the total number of variables in S. Let $a = (a_1, a_2, \cdots, a_l)$ be random variables such that all of them are yielded from either E or E', where l is polynomial in k.*

A distinguisher D for (E, E') is a probabilistic algorithm which halts in time polynomial in k with output in $\{0, 1\}$ and satisfies $(i)D(a, E) = 1$ iff a is from E; $(ii)D(a, E') = 1$ iff a is from E'.

If $Adv(D) = |Pr[D(a, E) = 1] - Pr[D(a, E') = 1]|$, then D distinguishes (E, E') with

advantage $Adv > 0$.

E, E' are said to be polynomially indistinguishable if there exists no distinguisher for (E, E') with advantage $Adv > 0$ non-negligible in k for all sufficiently large k.

A concept that is closely related to indistinguishability is the distance between two distributions. If the statistical distance between two distributions is negligible, these two distributions are indistinguishable. The formula for distance of two distributions is introduced.

Definition 2.2.5 (Statistical Indistinguishability) *Let $Pr_D(x)$ denote the probability of x in discrete probability distribution D. Then the distance between distributions D_1 and D_2 is*

$$\mathsf{dist}(D_1, D_2) = \frac{1}{2} \sum_x \left| Pr_{D_1}(x) - Pr_{D_2}(x) \right|.$$

Note that for any two distributions $\mathsf{dist}(D_1, D_2) \leq 1$. Two distributions D_1 and D_2 are statistically indistinguishable *if $\mathsf{dist}(D_1, D_2)$ is negligible.*

Note that if two distributions are statistically indistinguishable, then they are also computationally indistinguishable.

2.2.2 Formal Definition of Digital Signature Schemes

The digital signature concept was proposed by Diffie and Hellman in 1976 [14]. However, the formal mathematical definition that is standard today only appeared later, after refinement by several researchers. The formal definition for signature schemes due to Goldwasser *et al* [24] is introduced.

Definition 2.2.6 (Signature Schemes) *A signature scheme is a triple,* (Gen, Sig, Ver)*, of probabilistic polynomial time algorithms as defined below:*

- *Algorithm* $Gen(1^k)$ *is called the key generation algorithm. There exists a polynomial* $kl(\cdot)$*, called the key length, so that on input* 1^k*, algorithm Gen outputs a pair* (sk, vk) *so that* $sk, vk \in \{0,1\}^{kl(k)}$*. The first element,* sk*, is called the signing key, and the second element* vk *is the corresponding verification key.*

- *Algorithm* $Sig(sk, m)$ *is called the signing algorithm. There exists a polynomial* $ml(\cdot)$*, called the message length, so that on input a pair* (sk, m)*, where* $sk \in \{0,1\}^{kl(k)}$ *and* $m \in \{0,1\}^{ml(k)}$*, algorithm Sig outputs a string called a signature of message* m*.*

- *Algorithm* $Ver(vk, m, \sigma)$ *is called the verification algorithm that outputs either "accept" or "reject". For every* k*, every* (sk, vk) *in the range of* $Gen(1^k)$*, every* $m \in \{0,1\}^{ml(k)}$ *and every* σ *in the range of* $Sig(sk, m)$*, it holds that*

$$Ver(vk, m, \sigma) = accept.$$

The security properties of secure digital signature schemes are described in the notion of *existential unforgeability under adaptive chosen message attacks*, which was proposed by Goldwasser, Micali and Rivest [24]. This notion has been widely used in the design of digital signature schemes. In this dissertation, this notion is used to prove the security of the proposed schemes. The definition given here appears in [21].

Definition 2.2.7 (Secure Signatures [21]) *A signature scheme* (Gen, Sig, Ver) *is existentially unforgeable under adaptive chosen message attack if it is infeasible for a forger who*

only knows the public key, to produce a valid (message, signature) pair, even after obtaining polynomially many signatures on messages of its choice from the signer.

Formally, for every probabilistic polynomial time forger algorithm $\mathcal{F}$, there exists a negligible function $negl()$ such that

$$Pr\left[\begin{array}{l} \langle vk, sk\rangle \leftarrow Gen(1^k); \\ for\ i = 1 \ldots n \\ \qquad m_i \leftarrow \mathcal{F}(vk, m_1, \sigma_1, \ldots, m_{i-1}, \sigma_{i-1});\ \sigma_i \leftarrow Sig(sk, m_i); \\ \langle m, \sigma\rangle \leftarrow \mathcal{F}(vk, m_1, \sigma_1, \ldots, m_n, \sigma_n), \\ s.t.\ m \neq m_i\ for\ i = 1 \ldots n,\ and\ Ver(vk, m, \sigma) = accept \end{array}\right] = negl(k).$$

2.2.3 Cryptographic Hash Functions

Hash functions are commonly used in cryptographic constructions. A hash function maps arbitrary strings of finite length to binary strings of fixed length. For cryptographic purposes, a hash function should satisfy one or more of the following security properties defined by Damgard [13].

Definition 2.2.8 (Cryptographically Secure Hash Function) *A hash function h maps arbitrary strings of finite length to binary strings of fixed length l_h, which is polynomial in k (the security parameter):*

$$h : \{0,1\}^* \rightarrow \{0,1\}^{l_h}.$$

A cryptographically secure hash function should satisfy at least one of the following properties:

- Strong collision resistance*: No probabilistic polynomial time algorithm can find a pair* (x, x') *with* $x \neq x'$ *such that* $h(x) = h(x')$.

- Weak collision resistance*: For a given* x*, no probabilistic polynomial time algorithm can find an* $x' \neq x$ *such that* $h(x) = h(x')$.

- One-way*: For a given* c*, no probabilistic polynomial time algorithm can find an* x *such that* $c = h(x)$.

One of the proposed schemes in this dissertation will use another property of hash functions called "division intractability," which was introduced by Gennaro *et al.* [21]. Informally speaking, a hash function is division intractable if it is infeasible to find distinct inputs for this hash function such that the hash value of one input divides the product of hash values of all other inputs.

Definition 2.2.9 (Division Intractability [21]) *A hash function* h *is division intractable if it is infeasible to find distinct inputs* $x_1, \ldots, x_n, y$ *such that* $h(y)$ *divides the product of the* $h(x_i)$*'s.*

Formally, for every probabilistic polynomial time algorithm $\mathcal{A}$*, there exists a negligible function negl() such that*

$$Pr\left[\begin{array}{l} \mathcal{A}(h) = \langle x_1, \ldots, x_n, y\rangle \\ \quad s.t.\ y \neq x_i\ for\ i = 1 \ldots n, \\ \quad and\ h(y)\ divides\ \prod_{i=1}^{n} h(x_i) \end{array}\right] = negl(k).$$

2.2.4 Hash-Sign-Switch Paradigm

As described in Chapter 1, Shamir and Tauman proposed a "Hash-Sign-Switch" technique that can convert any signature scheme to an online/offline signature construction using a trapdoor hash function. The formal definition for this technique is introduced here.

Definition 2.2.10 (Hash-Sign-Switch Paradigm) *For security parameter k, recall that a signature scheme $S = (Gen, Sig, Ver)$ has a public verification key vk, and a private signing key sk. Similarly, a trapdoor hash function h has a public hash key hk, and a private trapdoor key tk. An online/offline signature scheme can be generated as follows.*

Key Generation. *Output public key (vk, hk), and private key (sk, tk). Let $M = \{0,1\}^{l_m}$ be the message space for the scheme, and $R = \{0,1\}^{l_r}$ be the space of random input for a trapdoor hash function.*

Signing Algorithm. *The signing procedure includes the standard two phases for an online/offline scheme.*

Offline Phase: *Compute $h(hk, m', r')$, where $(m', r') \in_R M \times R$, and compute $\sigma = Sig(sk, h(hk, m', r'))$.*

Online Phase: *For a message m, compute r such that $h(hk, m, r) = h(hk, m', r')$ Then the signature is (σ, r).*

Verification Algorithm. *To verify that (σ, r) is a valid signature on message m, compute $h(hk, m, r)$, and check that*

$$Ver(vk, h(hk, m, r), \sigma) = accept.$$

Using this Hash-Sign-Switch paradigm, the resulting online/offline signature scheme needs to manage the second keypair (hk, tk), and its signature contains the additional random value r. In addition, this technique adds significant computational overhead in the offline phase because a trapdoor hash function is in fact another set of public key operations.

2.3 Complexity Assumptions

This section reviews the formal definitions for complexity assumptions used in this dissertation. Specifically, the strong RSA assumption and its variant are used. The strong RSA assumption is a well-accepted complexity assumption in cryptography, which was first proposed by Baric and Pfitzmann [3] and Fujisaki and Okamoto [20].

Assumption 2.3.1 (Strong RSA Assumption) *Let n be an RSA modulus. The* flexible RSA problem *is the problem of taking a random element $u \in Z_n^*$ and finding a pair (v, e) such that $e > 1$ and $v^e = u \mod n$. The* strong RSA assumption *says that no probabilistic polynomial time algorithm can solve the flexible RSA problem for random inputs with non-negligible probability.*

The strong RSA assumption itself is defined over Z_n^* for when n is an RSA modulus. Cramer and Shoup investigated this assumption over QR_n for n being a special RSA modulus, and showed that the assumption is no stronger in QR_n than in general Z_n^*. Some of the proposed schemes use computation over QR_n and rely on this assumption.

When investigating cryptography over a small subgroup of Z_n^*, Groth also introduced a variant of the strong RSA assumption, called the strong RSA subgroup assumption, over a specially formed small subgroup of Z_n^* — this definition is given here, with terminology

slightly cleaned up from the original paper [25].

Assumption 2.3.2 (Strong RSA Subgroup Assumption) *Let K be a key generation algorithm that produces an RSA subgroup pair (n, g). The flexible RSA subgroup problem is to find $u, w \in Z_n^*$ and $d, e > 1$ such that $g = uw^e \mod n$ and $u^d = 1 \mod n$. The strong RSA subgroup assumption for this key generation algorithm states that it is infeasible to solve the flexible RSA subgroup problem with non-negligible probability for inputs generated by K.*

To facilitate the proofs for the proposed schemes in the subsequent chapters, a multiple generator version of the strong RSA subgroup assumption is presented, which is implied by a lemma in Groth's paper (Lemma 1 in [25]).

Assumption 2.3.3 (Strong RSA Subgroup Assumption with Multiple Generators) *Let K be a key generation algorithm that produces an RSA subgroup pair (n, g). Let $g_1, \ldots, g_w$ be randomly chosen generators of G. The multiple generator version of the flexible RSA subgroup problem is to find values $(y, e, e_1, \ldots, e_w)$ such that $y^e = g_1^{e_1} \ldots g_w^{e_w} \mod n$. If $e = 0$, then it is required that there exists an e_i for $i \in [1, w]$ such that $e_i \neq 0$. Otherwise there exists an e_i for $i \in [1, w]$ such that $GCD(e, e_i) < e$. The strong RSA subgroup assumption with multiple generators for this key generation algorithm states that it is infeasible to solve the multiple generator version of the flexible RSA subgroup problem with non-negligible probability for inputs generated by K.*

Note that this assumption can also be used with a single generator when $w = 1$, despite it is a multiple generator version of the strong RSA subgroup assumption.

2.4 Models of Computation

Chapter 1 briefly introduced the concept of models of computation, and implications for cryptographic constructions. Two sets of signature schemes have been devised in this dissertation, one of which is secure in the random oracle model, and the other of which is secure in the standard model. The standard model follows the standard definition of computational models in theory of computation, while the random oracle model is a special model in cryptography, which is described in this section.

2.4.1 Random Oracle Model

Many cryptographic constructions use a hash function to randomize values such as a message to be encrypted or signed. For example, in the hashed version of the RSA signature scheme [6], a message to be signed is hashed first, then the hash value in turn is signed by the RSA signing function.

The output of a cryptographically secure hash function "looks random" in all important ways when it can satisfy the security properties defined in 2.2.8. Therefore, the analysis can be simplified by assuming it *is* random. This analysis technique, in which the hash function is treated as a random function, was formalized by Bellare and Rogaway [4] as the so-called random oracle model, which treats the hash function as a random function publicly accessed by all participants in a scheme.

Definition 2.4.1 (Random Oracle) *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 given the last time.*

- *If it has not answered the query x before, it generates a random response which is uniformly chosen from the oracle's output range. The random oracle also records the answer for this x in case the same query is asked later.*

To carry out a security analysis or proof in the random oracle model, a public random oracle is set up to be accessed by all parities, either good or bad. Since random oracles are a mathematical convenience for the sake of analysis, when such an algorithm is implemented in practice the random oracle is replaced by a hash function. The random oracle methodology facilitates design and analysis of many cryptographic schemes. For example, the RSA scheme with the Optimal Asymmetric Encryption Padding (OAEP), which is one way the RSA scheme is used for encryption in practice, has been proved secure in the random oracle model [5, 40].

Unfortunately, in 1998, Canetti *et al.* constructed a scheme that can be proved secure in the random oracle model, while any real implementation will result in an insecure construction [10]. The basic idea is to start with a secure scheme and modify it to obtain a construction secure in the random oracle model, but this modified scheme does not enjoy any secure implementation. A special structure called an evasive relation is used in the modified scheme with the following operations: if a (query, answer) pair produced by the random oracle is in the evasive relation, output the private key; otherwise, the scheme behaves as the original construction. This modification implies that the scheme is broken if a query satisfying the evasive relation is made, since the private key is exposed. In the random oracle model, finding such a (query, answer) pair is infeasible so the modified scheme will maintain its security. However, it is easy to find such a pair under any implementation of the random

oracle, thus any implementation will not be secure.

Due to this fundamental flaw in the random oracle model, security proofs in the random oracle model do not necessarily imply security in the standard model, so a proof of security in the random oracle model can only be treated as (at best) a heuristic argument that a scheme is secure. However, people still use the random oracle model since it is possible to derive more efficient and simple algorithms this way. For example, most widely used constructions such as the RSA signature/encryption schemes can only be proved secure in the random oracle model. The lesson from Canetti *et al.*'s work is that we should always be somewhat skeptical about the security of schemes which can only be proved secure in the random oracle model.

CHAPTER 3

SIGNATURE SCHEMES IN THE RANDOM ORACLE MODEL

Since the strong RSA assumption was introduced in 1997, many new cryptographic constructions have been proposed (e.g., [1, 7, 8, 9, 12, 26]). This chapter introduces the core design technique used in this dissertation, generalizing the flexible RSA problem to use exponents on both sides of the equation, which is no easier to solve than the standard flexible RSA problem assuming certain conditions are met. The two-exponent version of the flexible RSA problem provides the flexibility of performing some computations before the message to be signed is known, and delaying other computations until the message is provided, allowing us to devise online/offline signature schemes. In this chapter, this design technique is described and used to devise two direct online/offline signature schemes which are secure in the random oracle model.

3.1 Design Technique

Informally, the strong RSA assumption states that when n is an RSA modulus, for a random $u \in Z_n^*$, no probabilistic polynomial time algorithm can find a pair $(v, e),\ e > 1$ such that

$$v^e = u \mod n,$$

with non-negligible probability. However, this problem is efficiently solvable if the factorization of n is known. Knowing the factorization of n, we can pick a random e relatively prime

to $\phi(n) = (p-1)(q-1)$, calculate d, the inverse of e modulo $\phi(n)$, and obtain $v = u^d \mod n$. This is exactly the process used by the RSA scheme. The only difference between the RSA problem and the flexible RSA problem is that e is a given input in the RSA problem while e is an output to be computed in the flexible RSA problem. This small difference has a big impact on the design of cryptographic constructions. The flexibility of the flexible RSA problem is one reason so many constructions have been proposed since the introduction of the strong RSA assumption.

In 1999 Gennaro, Halevi and Rabin proposed a signature scheme [21], which is called the GHR scheme, which produces a signature that satisfies

$$\sigma^{h(m)} = g \mod n,$$

where h is a hash function, g is a fixed integer in Z_n^*, and σ is the signature with respect to a message m. Notice that the signature σ and $h(m)$ give a solution to the flexible RSA problem, for input value $u = g$.

Now the two-exponent version of the flexible RSA problem is introduced. Let n be an special RSA modulus. The two-exponent version of the *flexible RSA problem* is the problem of taking a random element $u \in QR_n$ and finding a triple (v, e_1, e_2) such that $e_1 > 1, GCD(e_1, e_2) < e_1$, and $v^{e_1} = u^{e_2} \mod n$.

Notice, when e_2 is 1, the GCD condition is always met and we have exactly the standard flexible RSA problem. The following lemma, due to Camenisch and Lysyanskaya, shows that under the right conditions the two-exponent version of the flexible RSA problem is not any easier to solve than the standard flexible RSA problem [9].

Lemma 3.1.1 *Let n be a special RSA modulus. Given values $u, v \in QR_n$ and $x, y \in Z$, $GCD(x, y) < x$ such that $v^x \equiv u^y \mod n$. Values $z, w > 1$ such that $z^w \equiv u \mod n$ can be computed efficiently.*

Since a solution to the standard flexible RSA problem is also a solution to the two-exponent version (with $e_2 = 1$), and the preceding lemma shows that a solution to the two-exponent version can be used to compute a solution to the standard problem, the computational complexities of these two variants are equivalent. Furthermore, note that if the factorization of n is known we can compute more general solutions to the two-exponent version, with random exponents: pick two random numbers e_1, e_2, calculate e_1^{-1} with respect to $p'q'$, and obtain $v = u^{e_1^{-1}e_2} \mod n$.

However, notice that there exists another way to obtain a solution to the two-exponent version of the flexible RSA problem. We can first pick a random number r, and compute $v = u^r \mod n$. Then we pick an e_1, and compute $e_2 = r \times e_1 \mod p'q'$. This way, the computation of v is independent of e_1 and e_2, and e_2 is computed from r and e_1. The computation of v uses modular exponentiation, which is time consuming. Under this new method, v can be obtained independently of its exponent (which, in the signature schemes, includes the message to be signed), allowing us to devise online/offline signature schemes.

The GHR scheme fixes e_2 to be 1 and solves for v — but solving for v is slow and can't be done until m is known. The use of the two-exponent version allows us to pick v randomly and solve for e_2 after m is known, which is much more efficient.

3.2 The RQ Scheme

In this section, the first proposed direct online/offline signature scheme is presented, called the RQ scheme — recall that the name indicates that this scheme is designed for the *r*andom oracle model using computations over QR_n.

3.2.1 The Scheme

Now details of the online/offline design sketched above are given. When a message is known, the signer calculates the hash value for this message by using a division intractable hash function, which is modeled in the analysis by a random oracle. Then the techniques described in the previous section on the two-exponent flexible RSA problem are used in order to complete the signature. The formal description of the RQ scheme follows.

Public System Parameters. Let k be the security parameter. Let l_h be the output length of a hash function, which is polynomial in k, and should be chosen so that the failure probability in Lemma 3.2.5 is acceptably small.

Key Generation. On input 1^k, pick two k-bit safe primes p and q (so that $p = 2p' + 1$, and $q = 2q' + 1$, where p' and q' are also prime), and let $n = pq$. Let $l_n = 2k$ be the length of the public modulus used in the signing algorithm. Select b as a random generator of QR_n. Choose a division intractable hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$. Output private key $(p'q')$, and public key (n, b, h).

Signing Algorithm. The signing procedure includes two phases.

Offline Phase: Pick N_p random values $\gamma_i \in_R [0, p'q')$, for $i = 1, \cdots, N_p$. For each γ_i,

compute

$$v_i = b^{\gamma_i} \mod n.$$

Thus, the signer prepares a pool of N_p pairs (γ_i, v_i) in idle time.

ONLINE PHASE: Given a message $m \in [0, 2^{l_m})$, compute $h(m)$, then use the next unused (γ_i, v_i) pair to compute

$$s = \gamma_i \times h(m) \mod p'q'.$$

Optional test step: Test if $GCD(h(m), s) \leq 2^{2\sqrt{l_h}}$ — if so, the signature is ready; otherwise, the signer will continue by trying other (γ_i, v_i) pairs. The signature is (v_i, s) for the message m.

Verification Algorithm. To verify that (v, s) is a signature on message m, check that $GCD(h(m), s) \leq 2^{2\sqrt{l_h}}$, and

$$v^{h(m)} \equiv b^s \mod n.$$

The online phase of the signing algorithm includes an optional test step — if time can be taken to perform this step, the signing algorithm will never fail, but at the cost of a moderately expensive GCD computation. On the other hand, as shown in Lemma 3.2.5, the probability of this test failing is negligible, so the test can be omitted if a negligible failure probability is acceptable. If the test is omitted, the online phase only requires a single modular multiplication.

In a real application, it is desirable to have a pool of (γ_i, v_i) pairs available when needed. For instance, in the case of server authentication, the server might need a large pool to complete a large burst of authentication requests, so N_p could be large (e.g., over 10,000).

On the other hand, for a mobile device, N_p can be set reasonable small — a pool size of 5 or less should be enough for most situations. During idle time, the device produces new pairs to keep the pool full, or it could download new pairs when securely connected to a more powerful device.

3.2.2 Security Analysis

In this section, the security of the RQ scheme is discussed. The proof technique used is the "reduction to contradiction" method described in Chapter 1. The idea is as follows: An algorithm which uses a signature forgery algorithm to solve the flexible RSA problem, is created, where the proposed algorithm simulates the signature oracle for interaction with the forgery algorithm. First, the proposed algorithm for the flexible RSA problem is given an input (u, n) which is used to initialize the signature oracle simulator. Based on the (u,n) values, the signature oracle simulator sets up the signature parameters (including creating a simulated "key" for the signature scheme) and answers queries asked by an assumed signature forger. Under the notion of adaptive chosen message attack, the simulator should not make any assumption about the messages queries made by the signature forger. If, after polynomial times of queries, the forger can produce a new signature which is not answered by the simulator, then the simulator uses this forged signature to obtain a solution to the flexible RSA problem, which can only happen with negligible probability under the strong RSA assumption.

First, some lemmas which are needed for the security proof, are introduced. The first lemma addresses the smoothness of a random integer. A positive integer is called B-smooth if none of its prime factors is greater than B. For example, $84 = 2^2 \times 3 \times 7$, so 84 is a

7-smooth integer. The smoothness property for a random integer expressed in the following lemma will be used in the later proof. The proof for this lemma can be found in the proof of Lemma 6 presented by Gennaro *et al.* [21].

Lemma 3.2.1 *Let e be a random l_h-bit integer. The probability of e being $2^{2\sqrt{l_h}}$-smooth (i.e., all e's prime factors are no larger than $2^{2\sqrt{l_h}}$) is no larger than $2^{-2\sqrt{l_h}}$. In other words, the probability of e having at least one prime factor larger than $2^{2\sqrt{l_h}}$ is at least $1 - 2^{-2\sqrt{l_h}}$.*

The following lemma is used directly in the security proof for the RQ scheme — note that the condition on w is met for sufficiently large l_h whenever w is polynomial in k, which is in turn polynomial in l_h. The division intractability property of a hash function is based on this lemma: when a hash function outputs l_h-bit random integers for arbitrary inputs, it is intractable to find an input whose hash value can divide the product of other hash values. Due to the importance of this lemma, its proof is re-written to facilitate understanding of the subsequent proof. The original proof is presented by Gennaro *et al.* [21].

Lemma 3.2.2 *Let $e_1, e_2, \cdots, e_w$ be random l_h-bit integers, where $w \leq 2^{0.5\sqrt{l_h}}$. Let j be a randomly chosen index from $[1, w]$, and define $E = (\prod_{i=1}^{w} e_i)/e_j$. Then the probability that e_j divides E is less than $2^{-\sqrt{l_h}}$.*

Proof: We denote by smooth the event that e_j is $2^{2\sqrt{l_h}}$-smooth. From Lemma 3.2.1, we know that $Pr[\mathsf{smooth}] \leq 2^{-2\sqrt{l_h}}$.

Consider the case in which e_j is not $2^{2\sqrt{l_h}}$-smooth. Then e_j has at least one prime factor $p > 2^{2\sqrt{l_h}}$, so $Pr[e_j \text{ divides } E]$ is bounded by the probability that at least one of the e_i $(i \neq j)$ is divisible by p. Since the e_i's are chosen uniformly, the probability that any specific e_i is

divisible by p is at most $1/p < 2^{-2\sqrt{l_h}}$. Then, the probability that there exists an e_i which is divisible by p is at most $w \times 2^{-2\sqrt{l_h}}$, and based on the bound on w given in the lemma we get $w \times 2^{-2\sqrt{l_h}} < 2^{-1.5\sqrt{l_h}}$. Therefore, $Pr\,[p \text{ divides } E|\neg\mathsf{smooth}] < 2^{-1.5\sqrt{l_h}}$, and since p is a prime factor of e_j, we get $Pr[e_j \text{ divides } E|\ \neg\mathsf{smooth}] < 2^{-1.5\sqrt{l_h}}$.

Therefore, the probability that e_j divides E is at most

$$Pr[\mathsf{smooth}] + Pr[e_j \text{ divides } E|\neg\mathsf{smooth}] < 2^{-2\sqrt{l_h}} + 2^{-1.5\sqrt{l_h}} < 2^{-\sqrt{l_h}},$$

which completes the proof. □

Based on Lemma 3.2.2, another lemma is introduced, which is used directly in the security proof for the RQ scheme.

Lemma 3.2.3 *Let $e_1, e_2, \cdots, e_w$ be random l_h-bit integers, where $w \leq 2^{0.5\sqrt{l_h}}$. Let j be a randomly chosen index from $[1, w]$, and define $E = (\prod_{i=1}^{v} e_i)/e_j$. If s is an integer such that $GCD(e_j, s) \leq 2^{2\sqrt{l_h}}$, then the probability that e_j divides Es is less than $2^{-\sqrt{l_h}}$.*

Proof: We denote by smooth the event in which e_j is $2^{2\sqrt{l_h}}$-smooth. From Lemma 3.2.1, we know $Pr[\mathsf{smooth}] \leq 2^{-2\sqrt{l_h}}$.

Consider the case in which e_j is not $2^{2\sqrt{l_h}}$-smooth. Then e_j has at least one prime factor $p > 2^{2\sqrt{l_h}}$, thus, $Pr[e_j|Es]$ is bounded by the probability that at least one of the e_i $(i \neq j)$ or s is divisible by p. Since the e_i's are chosen uniformly, the probability that any specific e_i is divisible by p is at most $1/p < 2^{-2\sqrt{l_h}}$. Then, the probability that there exists an e_i which is divisible by p is at most $w \times 2^{-2\sqrt{l_h}}$, and based on the bound on w given in the lemma we get $w \times 2^{-2\sqrt{l_h}} < 2^{-1.5\sqrt{l_h}}$. Furthermore, since $GCD(e_j, s) \leq 2^{2\sqrt{l_h}}$, p cannot divide s due to

$p > 2^{2\sqrt{l_h}}$. Therefore, $Pr\left[p \text{ divides } Es | \neg \mathsf{smooth}\right] < 2^{-1.5\sqrt{l_h}}$, and since p is a prime factor of e_j, $Pr[e_j \text{ divides } Es | \neg \mathsf{smooth}] < 2^{-1.5\sqrt{l_h}}$.

Therefore, the probability that e_j divides Es is at most

$$Pr[\mathsf{smooth}] + Pr[e_j \text{ divides } Es | \neg \mathsf{smooth}] < 2^{-2\sqrt{l_h}} + 2^{-1.5\sqrt{l_h}} < 2^{-\sqrt{l_h}}.$$

□

The security proof uses the standard technique of simulating the signature oracle for a forgery algorithm, resulting in algorithm to solve an assumed hard problem (the flexible RSA problem in this instance). Unfortunately, we cannot perfectly simulate the signature oracle, because the distribution of the s value produced by the signing algorithm depends on the unknown value $p'q'$. However, we can simulate something very close to this distribution — in the main security theorem the following lemma is used to show that the "close" distribution is in fact close enough to establish the security of the RQ scheme.

Lemma 3.2.4 *Let p' and q' be as defined in the RQ scheme, so in particular p' and q' are $k-1$ bit prime numbers. Consider the following two distributions on pairs (e, s):*

- *Distribution D_1 is obtained by selecting e uniformly from the set of l_h-bit integers, and s is computed as $s = \gamma \times e \bmod p'q'$, where γ is chosen uniformly from $(0, \cdots, p'q' - 1)$.*
- *Distribution D_2 is obtained by selecting e uniformly from the set of l_h-bit integers, and s is uniformly selected from $(0, \cdots, \frac{n-1}{4} - 1)$.*

$\mathsf{dist}(D_1, D_2) < 2^{-k+3}$, *so distributions D_1 and D_2 are statistically indistinguishable.*

Proof: $Pr_{D_i,e}(s)$ is used to denote the probability distribution of s values under D_i conditioned on e being a specific, given value. If S is a set of e values, the notation $Pr_{D_i,S}(s)$ is used to denote the probability distribution of s values under D_i conditioned on e being in S.

Since e is chosen independently as a uniformly distributed l_h-bit integer in both D_1 and D_2, it makes sense to talk about the probability that $e \in S$ for some set S without specifying distribution D_1 or D_2, and this is denoted as $Pr[e \in S]$.

The possible e values are partitioned into two sets, S_1 and S_2, defined as follows:

$$S_1 = \{e | GCD(e, p'q') = 1\}$$

$$S_2 = \{e | GCD(e, p'q') > 1\}$$

Now note that the goal is to compute $2 \cdot \mathsf{dist}(D_1, D_2)$, or

$$\sum_{(s,e)} |Pr_{D_1}(s,e) - Pr_{D_2}(s,e)| =$$

$$Pr[e \in S_1] \sum_{(s,e)|e \in S_1} |Pr_{D_1,S_1}(s) - Pr_{D_2,S_1}(s)| +$$

$$Pr[e \in S_2] \sum_{(s,e)|e \in S_2} |Pr_{D_1,S_2}(s) - Pr_{D_2,S_2}(s)|$$

Intuitively, the situation is that when $e \in S_1$ the distance between D_1 and D_2 is negligible. However, when $e \in S_2$, the distributions are quite different, but the probability of $e \in S_2$ is negligible.

First, consider one particular $e \in S_1$, so e is relatively prime to $p'q'$, and hence for any $s \in \{0, \cdots, p'q' - 1\}$ there is exactly one $\gamma \in \{0, \cdots, p'q' - 1\}$ such that $s \equiv \gamma \times e \pmod{p'q'}$

— namely, $\gamma = s \times e^{-1} \bmod p'q'$, where e^{-1} is the inverse of e in $Z^*_{p'q'}$. This gives a one-to-one mapping between s and γ values in this case, and since γ is uniformly distributed we have that $Pr_{D_1,e}(s) = \frac{1}{p'q'}$ for this e and any $s \in \{0, \cdots, p'q' - 1\}$. Furthermore, since s is chosen uniformly in distribution D_2, we have $Pr_{D_2,e}(s) = \frac{4}{n-1}$ for any $s \in (0, \cdots, \frac{n-1}{4} - 1)$. Since $Pr_{D_1,e}(s) = 0$ whenever $p'q' \leq r < \frac{n-1}{4}$, we get

$$\sum_{(s,e)|e \in S_1} |Pr_{D_1,S_1}(s) - Pr_{D_2,S_1}(s)|$$

$$\begin{aligned}
&= \left(\frac{1}{p'q'} - \frac{4}{n-1}\right) p'q' + \frac{4}{n-1}\left(\frac{n-1}{4} - p'q'\right) \\
&= 2 - \frac{8p'q'}{n-1} \\
&= 2 - \frac{8p'q'}{4p'q' + 2p' + 2q'} \\
&= \frac{8p'q' + 4p' + 4q' - 8p'q'}{4p'q' + 2p' + 2q'} \\
&= \frac{2p' + 2q'}{2p'q' + p' + q'} \\
&< \frac{p' + q'}{p'q'} \\
&= \frac{1}{p'} + \frac{1}{q'}.
\end{aligned}$$

Next, we turn to the case where $e \in S_2$, and consider $Pr[e \in S_2]$. For e to be in S_2, it must be a multiple of p' or q' (or possibly both). For any l_h, the probability that a randomly chosen l_h-bit integer is a multiple of p' is at most $\frac{1}{p'}$, and the probability that e is a multiple of q' is at most $\frac{1}{q'}$. Therefore,

$$Pr[e \in S_2] \leq \frac{1}{p'} + \frac{1}{q'}.$$

Returning to the overall distance between distributions D_1 and D_2, we get

$$\sum_{(s,e)} |Pr_{D_1}(s,e) - Pr_{D_2}(s,e)|$$

$$\begin{aligned}
&= Pr[e \in S_1] \sum_{(s,e)|e \in S_1} |Pr_{D_1,S_1}(s) - Pr_{D_2,S_1}(s)| + \\
&\quad Pr[e \in S_2] \sum_{(s,e)|e \in S_2} |Pr_{D_1,S_2}(s) - Pr_{D_2,S_2}(s)| \\
&< \left(1 - \frac{1}{p'} - \frac{1}{q'}\right) \times \left(\frac{1}{p'} + \frac{1}{q'}\right) + \left(\frac{1}{p'} + \frac{1}{q'}\right) \times 1 \\
&< 1 \times \left(\frac{1}{p'} + \frac{1}{q'}\right) + \left(\frac{1}{p'} + \frac{1}{q'}\right) \times 1 \\
&= 2 \times \left(\frac{1}{p'} + \frac{1}{q'}\right).
\end{aligned}$$

Therefore, $\mathsf{dist}(D_1, D_2) < \left(\frac{1}{p'} + \frac{1}{q'}\right)$, and since p' and q' are $k-1$ bits, we conclude that $\mathsf{dist}(D_1, D_2) < 2^{-k+3}$. □

Lemma 3.2.5 *Let (e, s) be a pair drawn from D_1, where $l_h \geq 4$ is polynomial in k. Then the probability that $GCD(e, s) > 2^{2\sqrt{l_h}}$ is at most $\frac{l_h}{2^{2\sqrt{l_h}}} + \nu(k)$, where $\nu(k)$ is negligible in k.*

Proof: We first consider the probability that $GCD(e, s) > 2^{2\sqrt{l_h}}$ when (e, s) comes from D_2. If $GCD(e, s) > 2^{2\sqrt{l_h}}$ then e and s are both divisible by some $d > 2^{2\sqrt{l_h}}$, and the probability that e and s are both divisible by an integer d is no larger than $(\frac{1}{d})^2$. Then, the probability that $GCD(e, s) > 2^{2\sqrt{l_h}}$ is equal to the probability that e and s are both divisible by some $d > 2^{2\sqrt{l_h}}$, which is

$$\leq \left(\frac{1}{2^{2\sqrt{l_h}} + 1}\right)^2 + \left(\frac{1}{2^{2\sqrt{l_h}} + 2}\right)^2 + \cdots + \left(\frac{1}{2^{l_h}}\right)^2$$

$$
\begin{aligned}
&< \frac{1}{2^{2\sqrt{l_h}}}\left(\frac{1}{2^{2\sqrt{l_h}}+1}+\frac{1}{2^{2\sqrt{l_h}}+2}+\cdots+\frac{1}{2^{l_h}}\right)\\
&= \frac{1}{2^{2\sqrt{l_h}}}(H_{2^{l_h}}-H_{2^{2\sqrt{l_h}}})\\
&< \frac{1}{2^{2\sqrt{l_h}}}H_{2^{l_h}}\\
&< \frac{1}{2^{2\sqrt{l_h}}}\left(\ln 2^l_h+1\right)\\
&= \frac{l_h}{2^{2\sqrt{l_h}}}\left(\ln 2+\frac{1}{l_h}\right)\\
&< \frac{l_h}{2^{2\sqrt{l_h}}}.
\end{aligned}
$$

By Lemma 3.2.4, D_1 and D_2 are statistically indistinguishable, so the probability that $GCD(e,s) > 2^{2\sqrt{l_h}}$ when (e,s) comes from D_1 can be larger than $\frac{l_h}{2^{2\sqrt{l_h}}}$ by only a negligible amount. Therefore, $Pr[GCD(e,s) > 2^{2\sqrt{l_h}}] \leq \frac{l_h}{2^{2\sqrt{l_h}}} + \nu(k)$, where $\nu(k)$ is negligible in k. □

Now the proof for the RQ scheme is secure under the strong RSA assumption when the hash function h is replaced by a random oracle, is given as follows.

Theorem 3.2.1 *In the random oracle model, the RQ scheme is existentially unforgeable under an adaptive chosen message attack, assuming the strong RSA assumption.*

Proof: Let F be a forgery algorithm. Under the random oracle model, F always queries the random oracle about a message m before it either asks the signature oracle to sign this message, or outputs (m,v,s) as a potential forgery. Let w be some polynomial upper bound on the number of queries that F makes to the random oracle.

Now an efficient algorithm A is shown, that uses F as a subroutine, such that if F has probability ϵ of forging a signature, then A has probability $\epsilon' \approx \epsilon/w$ of solving the flexible RSA problem.

A is given a special RSA modulus n and a $t \in_R QR_n$, and its goal is to find a pair (z, e) such that $e > 1$ and $z^e \equiv t \mod n$.

First, A prepares answers for the random oracle queries that F will ask by picking w random l_h-bit integers $e_1, \cdots, e_w$ and a random $j \in_R [1, w]$. A is betting on the chance that F will use its j'th oracle query to generate the forgery.

Next, A prepares answers for signature queries that F will ask. A computes $E = (\prod_{i=1}^{w} e_i)/e_j$. If e_j divides E, then A outputs "failure" and halts. Otherwise, it sets $b = t^E \mod n$, and initializes the forger F, giving it the public key (n, b).

A then runs the forger algorithm F, answering oracle queries with the help of a function $\text{SAMPLE}_D()$ which gets a random sample from some distribution D described in Lemma 3.2.4. Specifically, oracle queries are answered as follows:

- Random oracle queries for $m_1 \ldots m_w$ are answered by setting $h(m_i) = e_i$ for each $i \in [1, w]$.

- Signature oracle queries for message m_i, for $i \neq j$, are answered with (m_i, v_i, s_i) where $s_i = \text{SAMPLE}_D()$ and $v_i = t^{Es_i/e_i} \mod n$.

If F queries the signature oracle for message m_j, or halts with an output other than (m_j, v_j, s_j) for which $v_j^{e_j} \equiv b^{s_j} \mod n$ and $GCD(e_j, s_j) \leq 2^{2\sqrt{l_h}}$, then A outputs "failure" and halts. Otherwise we have a valid forgery with

$$v_j^{e_j} \equiv b^{s_j} \equiv t^{Es_j} \mod n.$$

By Lemma 3.2.3, e_j divides Es_j with a negligible probability, and so with overwhelming

probability $GCD(e_j, Es_j) < e_j$, and we can apply Lemma 3.1.1 to find values z and e such that $z^e \equiv t \mod n$. Therefore, if A does not output "failure" then with overwhelming probability it outputs a valid solution to this instance of the flexible RSA problem.

To bound the probability that A outputs "failure", we need to examine the probability distribution D used in answering queries to the signature oracle. If we could sample according to distribution D_1, as defined in Lemma 3.2.4, then the distribution of signature responses is identical to that produced by an actual signer, so the probability of generating a valid forgery for a specific message is ϵ, and combined with the probability that we correctly picked the correct j for the forgery query the overall success probability is ϵ/w. However, we cannot sample from distribution D_1 without knowing the value $p'q'$, which is not available to us in A's simulation of the signature oracle.

Instead, we use distribution D_2 from Lemma 3.2.4, which is just the uniform distribution on $(0, \cdots, \frac{n-1}{4} - 1)$ and so can be efficiently sampled. The success probability of the forgery algorithm is only changed by a negligible amount, since as showed in Lemma 3.2.4 that these two distributions are statistically indistinguishable (if the change in success probability was more than a negligible amount, we could use this very simulation as a distinguisher between D_1 and D_2). In other words, using D_2 for the distribution D, the success probability of the forgery algorithm is at least $\epsilon - \eta(k)$, where $\eta(k)$ is some negligible function in k, so A's probability of success is at least $(\epsilon - \eta(k))/w$.

A can also output "failure" if, in selecting the random oracle outputs $e_1, \cdots, e_w$, we have Es_j divisible by e_j. However, Lemma 3.2.3 established that this is another negligible probability, and so the overall probability of successfully solving the flexible RSA problem is $\approx \epsilon/w$. □

3.3 The RG Scheme

The RQ scheme nicely implements direct online/offline signing, and is a clear example of the proposed design technique based on the two-exponent version of the flexible RSA problem. However, it is not a completely satisfactory construction in terms of efficiency since the scheme requires a GCD check, an optional step for the signer, but a required test for the verifier. This section introduces a new construction, called the RG scheme, which uses computation over a small subgroup G of Z_n^* to improve overall computing efficiency. Furthermore, the method of online/offline signing is adjusted, making the GCD check unnecessary in the RG scheme.

3.3.1 The Scheme

Public System Parameters. Let k be the security parameter. Let l_h be the output length of a hash function, which is polynomial in k. l is a security parameter that controls the statistical closeness of distributions, and should be polynomial in k (in practice $l = 60$ is sufficient).

Key Generation. On input 1^k, pick two k-bit primes p and q as in Definition 2.1.8 (so $p = 2p'r_p + 1$, and $q = 2q'r_q + 1$, where p' and q' are also prime), and let $n = pq$. Let $l_n = 2k$ be the length of the public modulus, let $l_{p'q'}$ be the length of $p'q'$, and let $l_s = l_{p'q'} + l$ be the length of a specific exponent used in the signing algorithm. Let G be the unique subgroup of Z_n^* of order $p'q'$, and select a random generator b of G. Select $\beta \in_R [0, p'q')$ and compute $c = b^\beta \mod n$. Choose a division intractable hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$. Let

$K = \lfloor 2^{l_s}/p'q' \rfloor$. Output public key (n, b, c, h), and private key $(p'q', \beta, K)$.

Signing Algorithm. The signing procedure includes two phases.

Offline Phase: The signer picks a random $\gamma \in_R [0, p'q')$, and a random $k' \in_R [0, K)$, and then computes

$$v = b^{\gamma} \mod n,\ \lambda = k'p'q' - \beta \mod Kp'q'.$$

Online Phase: When a message $m \in [0, 2^{l_m})$ appears, the signer computes

$$s = \lambda + \gamma \times h(m) \mod Kp'q'.$$

The signature is (v, s) for the message m.

Verification Algorithm. To verify that (v, s) is a signature on message m, check that

$$v^{h(m)} \equiv b^s c \mod n.$$

This new scheme is very efficient for the signer. Given reasonable parameters $l_n = 1024$, $l_{p'q'} = 200$, $l = 60$, and $l_s = 200 + 60 = 260$, the offline signing cost is about 200 modular multiplications, while the online signing needs one modular multiplication of 1024-bit and 200-bit numbers and an modular addition with a 260-bit modulus. The GCD check is not needed in this scheme.

3.3.2 Security Analysis

In this section, the RG scheme is proved secure under Groth's strong RSA subgroup assumption over the small subgroup G of Z_n^*. First, two lemmas which are needed for the main proof, are presented as follows.

The first lemma addresses the indistinguishability of distributions $[0, Kp'q')$ and $[0, 2^{l_s})$ for $K, p'q'$, and s as defined in the RG scheme.

Lemma 3.3.1 *Let $K = \lfloor 2^{l_s}/p'q' \rfloor$, where K is superpolynomial in the security parameter k. The uniform distribution over $[0, Kp'q')$ is statistically indistinguishable from the uniform distribution over $[0, 2^{l_s})$.*

Proof: Let distribution D_1 be the uniform distribution over $[0, Kp'q')$, and let distribution D_2 be the uniform distribution over $[0, 2^{l_s})$.

Doing the basic algebra, we get

$$\begin{aligned}
dist(D_1, D_2) &= \frac{1}{2}\sum_x |Pr_{D_1}(x) - Pr_{D_2}(x)| \\
&= \frac{1}{2}[(\frac{1}{Kp'q'} - \frac{1}{2^{l_s}})Kp'q' + \frac{1}{2^{l_s}}(2^{l_s} - Kp'q')] \\
&= 1 - \frac{Kp'q'}{2^{l_s}} \\
&= 1 - \frac{\lfloor 2^{l_s}/p'q' \rfloor p'q'}{2^{l_s}} \\
&< 1 - \frac{((2^{l_s} - p'q')/p'q')p'q'}{2^{l_s}} \\
&= 1 - (1 - \frac{p'q'}{2^{l_s}}) \\
&= \frac{p'q'}{2^{l_s}}.
\end{aligned}$$

Thus, the distance between D_1 and D_2 is less than $\frac{p'q'}{2^{l_s}}$, and since $\frac{2^{l_s}}{p'q'}$ is superpolynomial

in the security parameter k, this distance is negligible. So, the uniform distribution over $[0, Kp'q')$ is statistically indistinguishable from the uniform distribution over $[0, 2^{l_s})$. □

The second lemma shows s as used in the RG scheme is statistically indistinguishable from the uniform distribution over $[0, 2^{l_s})$.

Lemma 3.3.2 *Let $K = \lfloor 2^{l_s}/p'q' \rfloor$, where K is superpolynomial in the security parameter k. Let β be a constant in $[0, p'q')$, $k' \in_R [0, K)$, and $\gamma \in_R [0, p'q')$. If we define $s = (k'p'q' - \beta + \gamma h(m)) \bmod Kp'q'$, where $h : \{0,1\}^{l_m} \rightarrow \{0,1\}^{l_n}$ is modeled with a random oracle, then the distribution of s is statistically indistinguishable from the uniform distribution over $[0, 2^{l_s})$.*

Proof: First, we prove that $h(m)$ is relatively prime to $p'q'$ with overwhelming probability. For $h(m)$ is not relatively prime to $p'q'$, it must be a multiple of p' or q' (or possibly both). For any l_h, the probability that a randomly chosen l_h-bit integer $h(m)$ is a multiple of p' is at most $\frac{1}{p'}$, and the probability that $h(m)$ is a multiple of q' is at most $\frac{1}{q'}$. Therefore,

$$Pr[GCD(h(m), p'q') > 1] \leq \frac{1}{p'} + \frac{1}{q'}.$$

So, the probability that $h(m)$ is not relatively prime to $p'q'$ is negligible.

Now consider the case when $h(m)$ is relatively prime to $p'q'$. For any $x \in [0, Kp'q')$, since $h(m)$ is relatively prime to $p'q'$, there exists exactly one pair (k', γ) such that $x = (k'p'q' - \beta + \gamma h(m)) \bmod Kp'q'$. Therefore there is a one-to-one mapping between pairs (k', γ) and values in $[0, Kp'q')$, and since pairs (k', γ) are chosen uniformly, the resulting distribution of s over $[0, Kp'q')$ is uniform. According to Lemma 3.3.1, the distribution of s

is statistically indistinguishable from the uniform distribution over $[0, 2^{l_s})$. Combined with the negligible probability that $h(m)$ is not relatively prime to $p'q'$, we get the result stated in the lemma. □

Now the RG scheme is secure under the strong RSA subgroup assumption when the hash function h is replaced by a random oracle, is proved as follows.

Theorem 3.3.1 *In the random oracle model, the RG scheme is existentially unforgeable under an adaptive chosen message attack, assuming the strong RSA subgroup assumption.*

Proof: Let F be a forger algorithm. Under the random oracle model, F always queries the random oracle about a message m before it either asks the signature oracle to sign this message, or outputs (m, v, s) as a potential forgery. Let w be some polynomial upper bound on the number of queries that F makes to the random oracle.

We now show an efficient algorithm A for the multiple generator version of the strong RSA subgroup problem in Assumption 2.3.3, that uses F as a subroutine, such that if F has probability ϵ of forging a signature, then A has probability $\epsilon' \approx \epsilon/w$ of solving the multiple generator version of the strong RSA subgroup problem. In the following we show that such an A is impossible, so we conclude that no successful forger F can exist.

A is given a special RSA modulus n and a generator $t \in_R G$, and its goal is to find a solution to the multiple generator version of the strong RSA subgroup problem.

First, A prepares answers for the random oracle queries that F will ask by picking w random l_h-bit integers $e_1, \cdots, e_w$ and a random $j \in_R [1, w]$. A is betting on the chance that F will use its j'th oracle query to generate the forgery.

Next, A prepares answers for signature queries that F will ask. A computes $E =$

$(\prod_{i=1}^{w} e_i)/e_j$, and picks at random an l_s-bit long s. If e_j divides E, then A outputs "failure" and halts. Otherwise, it sets $b = t^E \mod n$, $c = b^s \mod n$ and initializes the forger F, giving it the public key (n, b, c).

A then runs the forger algorithm F, answering oracle queries as follows:

- Random oracle queries for $m_1 \ldots m_w$ are answered by setting $h(m_i) = e_i$ for each $i \in [1, w]$.
- Signature oracle queries for message m_i, for $i \neq j$, are answered with (m_i, v_i, s_i) where $s_i \in_R [0, 2^{l_s})$ and $v_i = t^{E(s_i+s)/e_i} \mod n$.

If F queries the signature oracle for message m_j, or halts with an output other than (m_j, v_j, s_j) for which $v_j^{e_j} \equiv b^{s_j} c \mod n$, then A outputs "failure" and halts. Otherwise we have a valid forgery with

$$v_j^{e_j} \equiv b^{s_j} c \equiv t^{E(s_j+s)} \mod n.$$

We can assume that e_j has a prime factor $\pi > 2^{2\sqrt{l_h}}$ and the probability that E contains π as a factor is negligible — this assumptions fail with negligible probability, due to Lemma 3.2.1 and Lemma 3.2.2. Next, we show that $(s_j + s)$ is divisible by π with a negligible probability.

Assume $(s_j + s)$ is divisible by π with non-negligible probability. Let $s = yp'q' + s'$, and note that the forger's view is independent of y. Therefore, if the forger succeeds for this value of s it must also succeed for a random $\widehat{s} = \widehat{y}p'q' + s'$ with $\widehat{y} \neq y$. Thus, π must divide $(s_j + s)$ when s is replaced by $\widehat{s}$, and so must divide the difference of these two values, leading to the requirement that π divides $(s - \widehat{s}) = (y - \widehat{y})p'q'$. However, the probability

that a random factor π divides $p'q'$ is negligible, and since y and $\widehat{y}$ are chosen randomly the probability that π divides $(y - \widehat{y})$ is also negligible. Thus, $(s_j + s)$ is divisible by π with a negligible probability.

As a result of these arguments, e_j divides $E(s_j + s)$ with a negligible probability, and so with overwhelming probability $GCD(e_j, E(s_j + s)) < e_j$. Therefore, if A does not output "failure" then with overwhelming probability it outputs a valid solution to this instance of the multiple generator version of the strong RSA subgroup problem. However, this is infeasible under the strong RSA subgroup assumption with multiple generators. We conclude that it is infeasible for F to forge a valid signature. □

3.4 Summary

In this chapter, two new direct online/offline digital signature schemes, which are proved secure under adaptive chosen message attack in the strong RSA assumption and the strong RSA subgroup assumption, respectively, have been presented.

First the two-exponent version of the flexible RSA problem was defined. Based on this new problem, the RQ scheme, an online/offline signature scheme in the random oracle model operating over QR_n, is devised, and the scheme is further improved by removing the GCD check and using a small subgroup G of Z_n^*, giving the second major result, which is called the RG scheme.

The RQ scheme is not a perfect construction. However, the core contribution for this scheme is that it implements direct online/offline signing under the strong RSA assumption. No constructions in the literature have done this before. The RQ scheme is the foundation for the further improvements. Indeed, based on the RQ scheme, the RG scheme, a simple

and efficient construction, is devised, which can accommodate most applications requiring online/offline signing.

However, one problem remains: the security proofs for these two schemes require the random oracle model. In the next chapter, some new techniques that remove this restriction are introduced, giving schemes that are secure in the standard model.

CHAPTER 4

SIGNATURE SCHEMES IN THE STANDARD MODEL

In Chapter 3, the basis of the new methods for online/offline signing, the two-exponent version of the flexible RSA problem, is introduced, and two online/offline signature schemes are designed. However, these two schemes can only be proved secure in the random oracle model. In this chapter, the new techniques for online/offline signing are extended, and a basic signature scheme which is proved secure in the standard model, is introduced. This result is further extended in two directions, for the two sample scenarios defined in Chapter 1: embedded device authentication, and bursty server authentication.

4.1 Design Technique

The proposed schemes in Chapter 3 are based on the two-exponent version of the flexible RSA problem, finding solutions to an equation of the form

$$v^{h(m)} = b^s \mod n.$$

The proof technique used for these techniques constructs a signing simulator by computing the generator b beforehand, such that $b = u^{r_1 \cdots r_w} \mod n$. Since the value b must be computed before any queries are made, but it depends on values $r_1 \cdots r_w$ derived from messages, the fundamental problem is that the message queries from the forger cannot be predicted. This demonstrates why random oracles are so powerful in analysis: since the

hash function produces a random value, independent of the actual value of m_i, it does not matter whether this random value is sampled before or after m_i is known. Therefore, the proof relies on the random oracle model to randomly assign a hash value for a message, i.e, $h(m_i) = r_i$.

To overcome this obstacle, we can take another approach when using the two-exponent version of the flexible RSA problem. We move m to the right of the equation, and construct an equation of the form

$$v^e = b^{m+s} \mod n,$$

where e and s are random values. Thus, a signing simulator can prepare b without relying on specific messages. Using this idea, we can design a scheme without requiring the random oracle model.

4.2 Basic Signature Scheme

In this section, a new direct online/offline signature scheme is presented, which is called the SQ scheme following the naming convention (secure in the *s*tandard model, and using the group QR_n where n is a special RSA modulus). Interestingly, the SQ scheme uses exactly the same verification formula as in the traditional (non-online/offline) scheme proposed by Camenisch and Lysyanskaya in 2002 [9], which is referred to as the "CL Scheme" in this dissertation. Technically, the SQ scheme could be viewed as an online/offline extension of the Camenisch-Lysyanskaya scheme. However, the new derivation was made independently, using a different set of design techniques and criteria. Furthermore, due to the different derivation, the new scheme provides much better performance than their construction, even

when offline and online phases are combined to use the SQ scheme as a traditional signature scheme.

The CL scheme produces a triple (v, e, s) as a signature, where e and s are chosen randomly, and v is computed from (e, s), the message to be signed, and the private key. The SQ scheme produces the same triple (v, e, s) for the signature. In the SQ scheme, v and e are chosen randomly and s is computed from (v, e), the message to be signed, and the private key (and with precomputation in the offline phase, computing s is much simpler than computing v). But since v is independent of e, s and the message to be signed, the selection of v can be done in the offline phase. It will be shown that the distribution of triples from the new algorithm is statistically indistinguishable from the distribution of triples from the CL scheme, so the new scheme enjoys the same strong security guarantees as the CL scheme. Since the new signing algorithm produces the same signature as in the CL scheme, the verification algorithm is the same as in the CL scheme.

4.2.1 The SQ Scheme

Public System Parameters. Let k be the security parameter, and define the following lengths: l_m is the length of the message to be signed, with the restriction that $l_m < k - 2$. l is a parameter that controls the statistical closeness of distributions, and should be at least polynomial in k (in practice $l = 160$ is sufficient). For convenience, some lengths are also defined based on these parameters: l_e is the length of an exponent in the signature algorithm, which satisfies $l_m + 2 \leq l_e < k$. $l_n = 2k$ is the length of the public modulus, and

$l_s = l_n + l_m + l$ is the length of another exponent used in the signing algorithm.

Key Generation. On input 1^k, pick two k-bit safe primes p and q (so that $p = 2p' + 1$, and $q = 2q' + 1$, where p' and q' are also prime), and let $n = pq$. Select b as a random generator of QR_n. Select $\alpha, \beta \in_R [0, p'q')$ and compute $a = b^\alpha \mod n$ and $c = b^\beta \mod n$. Let $K = \lfloor 2^{l_s}/p'q' \rfloor$. Output public key (n, a, b, c), and private key $(p'q', \alpha, \beta, K)$.

Signing Algorithm. The signing procedure includes two phases.

Offline Phase: The signer picks a random $\gamma \in_R [0, p'q')$, a random l_e-bit prime number e, and a random $k' \in_R [0, K)$, and then computes

$$v = b^\gamma \mod n,\ \lambda = k'p'q' + \gamma e - \beta \mod Kp'q'.$$

Online Phase: When a message $m \in [0, 2^{l_m})$ appears, the signer computes

$$s = \lambda - \alpha m \mod Kp'q'.$$

Note that while this is stated as a modular operation, the ranges of the values ensure that an adjustment to keep the value in range is only needed with negligible probability, and even then this is accomplished with a single addition. The signature is (v, e, s) for the message m.

Verification Algorithm. To verify that (v, e, s) is a signature on message m, check that e's length is l_e, and

$$v^e \equiv a^m b^s c \mod n. \tag{4.1}$$

It can be verified that a valid signature can always pass the verification test. Since these operations are being performed in QR_n, we consider operations in the exponent modulo $p'q'$ (the size of QR_n), and get

$$s \equiv \gamma e - \beta - \alpha m \pmod{p'q'},$$

and so

$$a^m b^s c \equiv b^{\alpha m + (\gamma e - \beta - \alpha m) + \beta} \equiv b^{\gamma e} \equiv v^e \mod n.$$

The salient characteristic for the signing algorithm is its online/offline mechanism. Most of the computation can be done before the appearance of a message, and the online phase only needs a single multiplication (where one of the values is short) and a subtraction.

4.2.2 Security Analysis

In this section, the proposed SQ scheme is compared with the CL scheme.

The SQ scheme produces signatures that are indistinguishable from those of the Camenisch-Lysyanskaya scheme. To see this, consider (4.1). In the CL scheme, a, b, c are randomly chosen from QR_n, and v is calculated as

$$v = (a^m b^s c)^{e^{-1}} \mod n, \tag{4.2}$$

where $s \in_R [0, 2^{l_s})$. In the SQ scheme, a, b, c are also random elements of QR_n, but s is calculated; however, the following lemma shows that s in SQ is statistically indistinguishable from uniform over $[0, 2^{l_s})$.

Lemma 4.2.1 *Let $K = \lfloor 2^{l_s}/p'q' \rfloor$, where K is superpolynomial in the security parameter k. Let e be a value that is relatively prime to $p'q'$, let α and β be constants in $[0, p'q')$, and let m be a constant in $[0, 2^{l_m})$. Let $k' \in_R [0, K)$ and $\gamma \in_R [0, p'q')$. If we define $s = (k'p'q' + \gamma e - \beta - \alpha m) \bmod Kp'q'$, then s is statistically indistinguishable from uniform over $[0, 2^{l_s})$.*

Proof: First, we prove that s is uniformly distributed over $[0, Kp'q')$. For any $s \in [0, Kp'q')$, since e is smaller than p' or q' (since $l_e < k$), it is relatively prime to $p'q'$ and so there exists exactly one pair (k', γ) such that $s = (k'p'q' + \gamma e - \beta - \alpha m) \bmod Kp'q'$. Therefore there is a one-to-one mapping between pairs (k', γ) and values in $[0, Kp'q')$, and since pairs (k', γ) are chosen uniformly, the resulting distribution of s over $[0, Kp'q')$ is uniform. Applying Lemma 3.3.1, we conclude that the distribution of s is statistically indistinguishable from the uniform over $[0, 2^{l_s})$. □

As a consequence of Lemma 4.2.1, the view of an attacker with respect to the SQ scheme is statistically indistinguishable from the view of an attacker with respect to the CL scheme, which gives the following lemma.

Lemma 4.2.2 *The view of an attacker with respect to the SQ scheme is statistically indistinguishable from the view of an attacker with respect to the CL scheme.*

Proof: Consider equation (4.1). In the CL scheme, a, b, c are randomly chosen from QR_n, and v is calculated as

$$v = (a^m b^s c)^{e^{-1}} \mod n, \tag{4.3}$$

where $s \in_R [0, 2^{l_s})$. In the SQ scheme, a, b, c are also random elements of QR_n, and in the proof of Lemma 4.2.1 it is shown that s in the SQ scheme is uniformly distributed over

$[0, Kp'q')$, which is statistically indistinguishable from $[0, 2^{l_s})$. Thus, the SQ scheme produces signatures that are statistically indistinguishable from those of the CL scheme. Therefore, the view of an attacker with respect to the SQ scheme is statistically indistinguishable from the view of an attacker with respect to the CL scheme. □

From the above lemma and the security proof given by Camenisch and Lysyanskaya [9], the following theorem can be given.

Theorem 4.2.1 *The SQ scheme is existentially unforgeable under an adaptive chosen message attack, assuming the strong RSA assumption.*

4.2.3 Performance Analysis

For concrete parameters, the recommended parameter settings from the CL scheme are used, with $k = 512$, so n is 1024 bits long. l_m can be chosen as 160, and messages longer than 160 bits can first be sent through a collision-resistant hash function (e.g., SHA-1) to produce a 160-bit message digest, which is then signed. As stated earlier, $l = 160$ is sufficient to ensure the statistical closeness of the signature's actual distribution to the simulated distribution in the proof of the scheme [9], so $l_s = 1024 + 160 + 160 = 1344$. For this setting of parameters, the cost of the signing algorithm is about 1022 modular multiplications and the generation of a 162-bit prime number in the offline phase, and one multiplication and one subtraction in the online phase. The SQ scheme avoids the multiplications related to s in the CL scheme, which is about 1344 modular multiplications. Furthermore, note that the SQ scheme does not require computation of the multiplicative inverse of e as required by the CL scheme (see (4.2)), so has advantages even when not used in the online/offline mode.

Table 4.1 shows the comparison of the CL and SQ schemes' signing cost when the key size is 1024 bits, where $m(n_1, n_2)$ is the time needed for one n_1-bit number by n_2-bit number multiplication, $add()$ is the time needed for one addition, $mm(n)$ is the time for an n-bit modular multiplication, $pg(n)$ is the time needed to generate a n-bit prime number, $inv()$ is the time needed to compute a multiplicative inverse of a number.

	Signing Algorithm	
	Offline Phase	Online Phase
CL	–	$2528 * mm(1024) + pg(162) + inv()$
SQ	$1022 * mm(1024) + pg(162)$	$m(160, 1022) + add()$

Table 4.1: Comparison of the CL and SQ schemes

4.3 Signature Scheme for Embedded Device Authentication

The SQ scheme can accommodate many application scenarios when online/offline signing is needed. However, it is possible to improve the efficiency further as long as the users are willing to accept the strong RSA subgroup assumption. In this section, techniques are investigated to reduce the offline cost, which is particularly important in the scenario defined in Chapter 1 for embedded device authentication.

With typical parameter settings, the main costs of the offline phase in the SQ scheme are due to an exponentiation taking 1022 modular multiplications and the generation of a 162-bit prime e. Using computation over a small subgroup G of Z_n^*, the exponentiation cost can be reduced from 1022 to 200 modular multiplications. To reduce the cost of generating a prime e, a division intractable hash function as in the RQ and RG schemes is used. Thus, a scheme specialized for embedded device authentication is obtained, which is called the SGE

scheme.

4.3.1 The SGE Scheme

This section introduces how to avoid using prime numbers explicitly for the exponent e. In the SQ signature scheme, there were two important system requirements: that $e \geq 2^{l_m+2}$, and e should not be chosen more than once. If we can somehow generate a random integer that always has a prime factor greater than m, we don't have to use a prime number explicitly. In order to accomplish this, a division intractable hash function is used to produce a pseudo-random hash value, the same technique that was used in Chapter 3. An l_h-bit integer of this type (for sufficiently large l_h) has at least one prime factor greater than $2^{2\sqrt{l_h}}$ with overwhelming probability by Lemma 3.2.1. For example, suppose $l_h = 1024$, then this prime factor is greater than 2^{64}. Unfortunately, this bound is too small compared to the message size which is set $l_m = 160$. To overcome this obstacle, the m can be split into three pieces as $m = m_1||m_2||m_3$ where "||" represents string concatenation, and the bit length of each sub-message is shorter than 63 bits. For simplicity of notation in this section, the message are split into three pieces as just described, but clearly this generalizes to other numbers of pieces.

Public System Parameters. Let k be the security parameter, and define the following lengths: l_m is the length of the message to be signed, with the restriction that $l_m < k - 2$. l is a parameter that controls the statistical closeness of distributions, and should be at least polynomial in k (in practice $l = 120$ is sufficient). For convenience, some lengths are also defined based on these parameters: $l_n = 2k$ is the length of the public modulus, $l_{p'q'}$ is the

length of $p'q'$, and $l_s = l_{p'q'} + l_m/3 + l$ is the length of another exponent used in the signing algorithm. A length l_h is defined to be the length of message digests produced by a division intractable hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$, with the requirement that $2\sqrt{l_h} \geq l_m/3 + 2$.

Key Generation. On input 1^k, pick two k-bit primes p, q such that $p = 2p'r_p + 1$, and $q = 2q'r_q + 1$, where p' and q' are also prime. Let $n = pq$, and let G be the unique subgroup of Z_n^* of order $p'q'$. Select a random generator b of G, select $\alpha_1, \alpha_2, \alpha_3, \beta \in_R [0, p'q')$, and define

$$a_1 = b^{\alpha_1} \bmod n, \quad a_2 = b^{\alpha_2} \bmod n, \quad a_3 = b^{\alpha_3} \bmod n, \quad c = b^{\beta} \bmod n.$$

Finally, let $K = \lfloor 2^{l_s}/p'q' \rfloor$. The public key is (n, a_1, a_2, a_3, b, c), while the private key is $(p'q', \alpha_1, \alpha_2, \alpha_3, \beta, K)$.

Signing Algorithm. The signing procedure includes two phases.

Offline Phase: Pick a random $\gamma \in_R [0, p'q')$, a random $r \in_R [0, 2^{l_r})$, and a random $k' \in_R [0, K)$. Compute

$$v = b^{\gamma} \mod n, \ \lambda = k'p'q' + \gamma \times h(r) - \beta \mod Kp'q'.$$

Online Phase: For $m \in [0, 2^{l_m})$, break m into pieces such that $m = m_1||m_2||m_3$ and the length of each piece is at most $\lceil l_m/3 \rceil$ bits. Compute

$$s = \lambda - \alpha_1 \times m_1 - \alpha_2 \times m_2 - \alpha_3 \times m_3 \mod Kp'q'.$$

The signature is (v, r, s).

Verification Algorithm. To verify that (v, r, s) is a signature on message m, check that r's length is l_r, and

$$v^{h(r)} \equiv a_1^{m_1} a_2^{m_2} a_3^{m_3} b^s c \mod n.$$

This new scheme is extremely efficient for the signer. Given parameters $l_n = 1024$, $l_{p'q'} = 200$, $l_h = 1024$, $l_r = 256$, $l_m = 180$, $l = 120$, and $l_s = 200 + 180/3 + 120 = 380$, the offline signing cost is about 200 modular multiplications, while the online signing needs three multiplications with small numbers as well as three subtractions.

4.3.2 Security Analysis

To prove the security of the SGE scheme, the multiple generator version of the strong RSA subgroup assumption defined in Chapter 2 is used.

Theorem 4.3.1 *The SGE scheme is existentially unforgeable under an adaptive chosen message attack, assuming the strong RSA subgroup assumption.*

Proof: Suppose there exists a probabilistic polynomial time forgery algorithm $\mathcal{F}$, which can launch an adaptive chosen message attack on SGE and output a valid signature which has not been produced by the signing oracle. Then we can construct a probabilistic polynomial time algorithm $\mathcal{A}$ for the multiple generator version of the strong RSA subgroup problem, defined in Assumption 2.3.3. $\mathcal{A}$ takes a random input (n, g_1, g_2, g_3), with generators $g_1, g_2, g_3 \in_R G$, and uses $\mathcal{F}$ as a subroutine. In the following proof, all exponentiations are done modulo n.

Suppose $\mathcal{F}$ asks w signature queries for messages $m_1, \ldots, m_w$, and $\mathcal{F}$ obtains signatures (v_1, r_1, s_1), ..., (v_w, r_w, s_w) before forging a valid signature (v, r, s) for a message m. We can

define three types of forgeries.

Type I: For all $1 \leq i \leq w, r \neq r_i$.

Type II: For some $1 \leq i \leq w, r = r_i, v = v_i$.

Type III: For some $1 \leq i \leq w, r = r_i, v \neq v_i$.

Any forgery algorithm which succeeds in producing forgeries with non-negligible probability must produce forgeries of at least one of these types with non-negligible probability. Next, we show how to construct three different algorithms for $\mathcal{A}$ such that if $\mathcal{F}$ succeeds in producing a forgery of a particular type, then the corresponding $\mathcal{A}$ will succeed in solving the multiple generator version of the strong RSA subgroup problem. By the strong RSA subgroup assumption, such an $\mathcal{A}$ is impossible, so we conclude that no successful forger $\mathcal{F}$ can exist.

Type I: For all $1 \leq i \leq w, r \neq r_i$. $\mathcal{A}$ works as follows: choose according to the signature algorithm distinct l_r-bit integers $r_1, \ldots, r_w$. Set $E = \prod_{i=1}^{w} h(r_i)$. $\mathcal{A}$ selects $t_1, t_2 \in_R [0, 2^{l_s})$, and sets $a_1 = g_1^E$, $a_2 = a_1^{t_1}$, $a_3 = a_1^{t_2}$, $b = g_2^E$, $c = g_3^E$. $\mathcal{A}$ gives (n, a_1, a_2, a_3, b, c) to the forger $\mathcal{F}$. $\mathcal{A}$ can answer the forger $\mathcal{F}$'s signature query $m_i = m_{i1}||m_{i2}||m_{i3}$ by choosing $s_i \in_R [0, 2^{l_s})$ and computing

$$v_i = g_1^{(m_{i1}+t_1 m_{i2}+t_2 m_{i3})E/h(r_i)} g_2^{s_i E/h(r_i)} g_3^{E/h(r_i)} = (a_1^{m_{i1}} a_2^{m_{i2}} a_3^{m_{i3}} b^{s_i} c)^{h(r_i)^{-1}}.$$

$\mathcal{A}$ gives $\mathcal{F}$ the signature (v_i, r_i, s_i) for message m_i, which is statistically indistinguishable from the signature produced by SGE.

Consider that the forger $\mathcal{F}$ outputs (v, r, s) for a message m, so we have

$$v^{h(r)} = a_1^{m_1} a_2^{m_2} a_3^{m_3} b^s c = g_1^{(m_1 + t_1 m_2 + t_2 m_3)E} g_2^{sE} g_3^E.$$

By Lemma 3.2.2, $h(r)$ divides E with a negligible probability, and so with overwhelming probability $GCD(h(r), E) < h(r)$. Therefore, $\mathcal{A}$ outputs a valid solution to this instance of the multiple generator version of the strong RSA subgroup problem with overwhelming probability. However, this is infeasible under the strong RSA subgroup assumption with multiple generators. We conclude that the $\mathcal{F}$ cannot produce a Type I forgery with non-negligible probability.

Type II: For some $1 \leq i \leq w, r = r_i, v = v_i$. $\mathcal{A}$ uses the same method as in Type I to prepare E, a_1, a_2, a_3, b, c, and answers the forger $\mathcal{F}$'s signature queries.

Consider now that the forger's signature is (v_i, r_i, s) on message m. We have

$$a_1^{m_{i1}} a_2^{m_{i2}} a_2^{m_{i3}} b^{s_i} = a_1^{m_1} a_2^{m_2} a_3^{m_3} b^s,$$

and so $a_1^{m_{i1} - m_1} a_2^{m_{i2} - m_2} a_3^{m_{i3} - m_3} b^{s_i - s} = 1$, which gives

$$g_1^{E((m_{i1} - m_1) + t_1(m_{i2} - m_2) + t_2(m_{i3} - m_3))} g_2^{E(s_i - s)} = 1 = y^0$$

for any non-zero y. Since $m_i \neq m$, $E((m_{i1} - m_1) + t_1(m_{i2} - m_2) + t_2(m_{i3} - m_3)) \neq 0$. Therefore, $\mathcal{A}$ outputs a valid solution to this instance of the multiple generator version of the strong RSA subgroup problem with overwhelming probability. However, this is infeasible

under the strong RSA subgroup assumption with multiple generators. We conclude that the $\mathcal{F}$ cannot produce a Type II forgery with non-negligible probability.

Type III: For some $1 \leq i \leq w, r = r_i, v \neq v_i$. $\mathcal{A}$ guesses the forger $\mathcal{F}$ will make the forgery by reusing r_i. $\mathcal{A}$ prepares E as in Type I, and picks at random an l_s-bit long t, and $(l_s - l_m/3 - l/2)$-bit long t_1, t_2, t_3, t_4. Then $\mathcal{A}$ sets up

$$b = g_2^{E/h(r_i)},\ a_1 = b^{t_1},\ a_2 = b^{t_2},\ a_3 = b^{t_3},\ c = b^{h(r_i)t_4 - t}.$$

$\mathcal{A}$ can answer all queries $j \neq i$ as in Type I. For query i, $\mathcal{A}$ computes $s_i = t - t_1 m_{i1} - t_2 m_{i2} - t_3 m_{i3}$, $v_i = b^{t_4}$ such that (v_i, r_i, s_i) is also a valid signature. Due to length restrictions over t, t_1, t_2, and t_3, the distribution of s_i is statistically indistinguishable from the uniform distribution over $[0, 2^{l_s})$, which is in turn indistinguishable from the distribution of signatures produced by SGE.

Consider now that the forgery $\mathcal{F}$ outputs a new signature (v, r_i, s) on message m. That is, $v^{h(r_i)} = a_1^{m_1} a_2^{m_2} a_3^{m_3} b^s c$. We can obtain

$$(v/v_i)^{h(r_i)} = g_2^{((m_1 - m_{i1})t_1 + (m_2 - m_{i2})t_2 + (m_3 - m_{i3})t_3 + (s - s_i))E/h(r_i)}.$$

We can assume that $h(r_i)$ has a prime factor $\pi > 2^{2\sqrt{l_h}}$ and the probability that $E/h(r_i)$ contains π as a factor is negligible — this assumptions fail with negligible probability, due to Lemma 3.2.1 and Lemma 3.2.2. Next, we show that

$$((m_1 - m_{i1})t_1 + (m_2 - m_{i2})t_2 + (m_3 - m_{i3})t_3 + (s - s_i)) \tag{4.4}$$

is divisible by π with a negligible probability.

Assume (4.4) is divisible by π with non-negligible probability. Let $t_1 = x_1 p'q' + t'_1$, and note that the forger's view is independent of x_1. Therefore, if the forger succeeds for this value of t_1 it must also succeed for a random $\widehat{t_1} = \widehat{x_1} p'q' + t'_1$ with $\widehat{x_1} \neq x_1$. Thus, π must divide (4.4) when t_1 is replaced by $\widehat{t_1}$, and so must divide the difference of these two values, leading to the requirement that π divides $(m_1 - m_{i1})(t_1 - \widehat{t_1}) = (m_1 - m_{i1})(x_1 - \widehat{x_1})p'q'$. However, since $2\sqrt{l_h} \geq l_m/3 + 2$, π cannot divide $m_1 - m_{i1}$. Furthermore, the probability that a random factor π divides $p'q'$ is negligible, and since x_1 and $\widehat{x_1}$ are chosen randomly the probability that π divides $(x_1 - \widehat{x_1})$ is also negligible. Thus, (4.4) is divisible by π with a negligible probability.

So, $h(r_i)$ divides (4.4) with a negligible probability, and so with overwhelming probability $GCD(e_j, E(s_j + s)) < e_j$. Therefore, $\mathcal{A}$ outputs a valid solution to this instance of the multiple generator version of the strong RSA subgroup problem with overwhelming probability. However, this is infeasible under the strong RSA subgroup assumption with multiple generators. We conclude that the $\mathcal{F}$ cannot produce a Type III forgery with non-negligible probability. □

4.4 Signature Scheme for Bursty Server Authentication

The scheme in the previous section focuses on reducing the offline computation cost. Now, investigate possible ways to reduce the online cost. As described in Chapter 1, online performance is extremely important for server authentication when bursty authentication requests come in simultaneously from remote applications.

In this section, a new construction, called the SGS scheme with the last letter S representing *s*erver, which has the best online performance of any currently-known online/offline signature scheme, is presented. The Shamir-Tauman trapdoor function is extended to a small subgroup G of Z_n^*, and online/offline signing without using "Hash-Sign-Switch" paradigm is directly implemented. The new scheme requires the modular reduction of a 420-bit value by a 260-bit modulus as well as one or two additions in the online phase when the key size is 1024 bits. Using the trapdoor hash proposed by Shamir and Tauman, the Shamir-Tauman construction requires the modular reduction of a 1184-bit value by a 1024-bit modulus as well as one or two additions in the online phase.

4.4.1 The Shamir-Tauman Trapdoor Hash Function

Shamir and Tauman proposed a trapdoor hash function whose security is based on the factoring assumption when the modulus n is a special RSA modulus [39]. The online computation for their hash function is very efficient, which only incurs one modular addition and a modular reduction. The following describes the Shamir-Tauman trapdoor hash function for reference.

Key Generation Algorithm *Gen*. On input 1^k, pick two k-bit safe primes p and q (so $p = 2p' + 1$, and $q = 2q' + 1$, where p' and q' are also prime), and let $n = pq$. Let $l_n = 2k$ be the length of the public modulus, and let l_m be the length of the message to be signed. Choose a random $g \in_R Z_n^*$ of order $\lambda(n) = lcm((p-1),(q-1)) = 2p'q'$. Output public hash key $hk = (n, g)$, and private trapdoor key $tk = (p, q)$.

Hash Function h. For $hk = (n, g)$, hash function is defined as $h(hk, m, r) = g^{m||r} \mod n$,

where m is the message to be signed, and r is a random number in $Z_{\lambda(n)}$.

Trapdoor Collision Computation. Given a pair $(m', r') \in Z_n \times Z_{\lambda(n)}$, and $m \in Z_n$, we need to find $r \in Z_{\lambda(n)}$ such that $g^{m||r} = g^{m'||r'} \mod n$, that is $2^k m' + r' = 2^k m + r \mod \lambda(n)$. Given the private trapdoor key $tk = (p, q)$, we can obtain r as

$$r = 2^k(m' - m) + r' \mod \lambda(n).$$

Computing $2^k(m' - m)$ is a simple add and shift operation. When $l_n = 1024$, $l_m = 160$, the reduction of the 1184 bit result modulo a 1024 bit modulus is about 6 times faster than a standard reduction of a 2048 bit product modulo a 1024 bit modulus. Shamir and Tauman estimate that software implementations of this collision finding procedure will be about 10 times faster than performing a single modular multiplication of two 1024 bit numbers.

4.4.2 The SGS Scheme

The verification algorithm in the SQ scheme takes the form of

$$v^e = a^m b^s c \mod n.$$

If we treat $m||r$ in the Shamir-Tauman hash function as a message in the SQ scheme, we can take advantage of the great online performance in their construction in a direct online/offline scheme, avoiding the need for extra keys and longer signatures as in the Shamir-Tauman scheme. The design of the SGS scheme follows from this observation, and next this scheme

in detail is described.

Public System Parameters. Let k be the security parameter, and define the following lengths: l_m is the length of the message to be signed, with the restriction that $l_m < k - 2$. l is a parameter that controls the statistical closeness of distributions, and should be at least polynomial in k (in practice $l = 60$ is sufficient). For convenience, some lengths are also defined based on these parameters: $l_n = 2k$ is the length of the public modulus, $l_{p'q'}$ is the size of the subgroup used, $l_r = l_{p'q'} + l$, t and l_e are subject to $t \times l_e \geq l_m + l_r + 2$.

Key Generation. On input 1^k, pick two k-bit primes p and q (so $p = 2p'r_p + 1$, and $q = 2q'r_q + 1$, where p' and q' are also prime, each with length $l_{p'q'}/2$), and let $n = pq$. Let G be the unique subgroup of Z_n^* of order $p'q'$, and select a random generator a of G. Select $\alpha, \beta \in_R [0, p'q')$ such that $b = a^\alpha \mod n$, and $c = a^\beta \mod n$. Let $K = \lfloor 2^{l_r}/p'q' \rfloor$. Output public key (n, a, b, c, t), and private key $(p'q', \alpha, \beta, K)$.

Signing Algorithm. The signing procedure includes two phases.

Offline Phase: Pick a random prime e with length l_e, a random $m' \in_R [0, 2^{l_m})$, a random $r' \in_R [0, 2^{l_r})$, and a random $s \in_R [1, e^t)$. Compute γ by

$$\gamma \times e^t = m'||r' + \alpha \times s + \beta \mod p'q'.$$

Then compute

$$v = a^\gamma \mod n.$$

ONLINE PHASE: For $m \in [0, 2^{l_m})$, compute

$$r = 2^{l_r}(m' - m) + r' \mod Kp'q'.$$

The signature is (v, e, s, r).

Verification Algorithm. To verify that (v, e, s, r) is a signature on message m, check that e's length is l_e, and

$$v^{e^t} \equiv a^{m||r} b^s c \mod n.$$

For concrete parameters, let $k = 512$, so n is 1024 bits long. l_m can be chosen as 160, and messages longer than 160 bits can first be sent through a collision-resistant hash function (e.g., SHA-1) to produce a 160-bit message digest, which is then signed. As stated earlier, $l = 60$ is sufficient to ensure the statistical closeness of the signature's actual distribution to the simulated distribution in the proof. Further other system parameters are set up as $l_{p'q'} = 200$, $l_e = 88$, $t = 5$, so $l_r = 200 + 60 = 260$. For this setting of parameters, the cost of the signing algorithm is about 200 modular multiplications and the generation of a 88-bit prime number in the offline phase, and a modular reduction of a 420-bit value by a 260-bit modulus as well as one or two additions in the online phase when the key size is 1024 bits.

4.4.3 Security Analysis

Theorem 4.4.1 *The SGS scheme is existentially unforgeable under an adaptive chosen message attack, assuming the strong RSA subgroup assumption.*

Proof: Suppose there exists a probabilistic polynomial time forgery algorithm $\mathcal{F}$, which can

launch an adaptive chosen message attack on the above signature scheme and output a valid signature which has not been produced by the signing algorithm. Then we can construct a probabilistic polynomial time algorithm $\mathcal{A}$ for the multiple generator version of the strong RSA subgroup problem. $\mathcal{A}$ takes a random input (n, g_1, g_2, g_3), with $g_1, g_2, g_3 \in_R G$, and uses $\mathcal{F}$ as a subroutine. In the following proof, all exponentiations are done modulo n.

Suppose $\mathcal{F}$ asks w signature queries for messages $m_1, \ldots, m_w$, and $\mathcal{F}$ obtains signatures (v_1, e_1, s_1, r_1), ..., (v_w, e_w, s_w, r_w) before forging a valid signature (v, e, s, r) for a message m. We can define three types of forgeries.

Type I: For all $1 \leq i \leq w, e \neq e_i$.

Type II: For some $1 \leq i \leq w, e = e_i, s = s_i$.

Type III: For some $1 \leq i \leq w, e = e_i, s \neq s_i$.

Any forgery algorithm which succeeds in producing forgeries with non-negligible probability must produce forgeries of at least one of these types with non-negligible probability. Next, we show how to construct three different algorithms for $\mathcal{A}$ such that if $\mathcal{F}$ succeeds in producing a forgery of a particular type, then the corresponding $\mathcal{A}$ will succeed in solving the multiple generator version of the strong RSA subgroup problem. By the strong RSA subgroup assumption, such an $\mathcal{A}$ is impossible, so we conclude that no successful forger $\mathcal{F}$ can exist.

Type I: For all $1 \leq i \leq w, e \neq e_i$. $\mathcal{A}$ works as follows: randomly choose according to the signature algorithm distinct l_e-bit primes $e_1, \ldots, e_w$. Set $E = \prod_{i=1}^{w} e_i^t$. $\mathcal{A}$ sets $a = g_1^E$, $b = g_2^E$, $c = g_3^E$. $\mathcal{A}$ gives (n, a, b, c) to the forger $\mathcal{F}$. $\mathcal{A}$ can answer the forger $\mathcal{F}$'s signature query m_i

by choosing $r_i \in_R [0, 2^{l_r})$, $s_i \in_R [1, e_i^t)$, and computing

$$v_i = g_1^{(m_i||r_i)E/e_i^t} g_2^{s_i E/e_i^t} g_3^{E/e_i^t} = (a^{m_i||r_i} b^{s_i} c)^{(e_i^t)^{-1}}.$$

$\mathcal{A}$ gives $\mathcal{F}$ the signature (v_i, e_i, s_i, r_i) for message m_i, which is statistically indistinguishable from the signature produced by the signature scheme.

Consider that the forger $\mathcal{F}$ outputs (v, e, s, r) for a message m, so we have

$$v^{e^t} = a^{m||r} b^s c = g_1^{(m||r)E} g_2^{sE} g_3^E.$$

By Lemma 3.2.2, e^t divides E with a negligible probability, and so with overwhelming probability $GCD(e^t, E) < e^t$. Therefore, $\mathcal{A}$ outputs a valid solution to this instance of the multiple generator version of the strong RSA subgroup problem with overwhelming probability. However, this is infeasible under the strong RSA subgroup assumption with multiple generators. We conclude that the $\mathcal{F}$ cannot produce a Type I forgery with non-negligible probability.

Type II: For some $1 \leq i \leq w, e = e_i, s = s_i$. $\mathcal{A}$ guesses the forger $\mathcal{F}$ will make the forgery by reusing e_i and s_i. $\mathcal{A}$ prepares E as in Type I, and picks at random $s' \in_R [1, e_i^t + 2^{l_m + l_r})$. Then set up

$$a = g_1^{E/e_i^t}, b = a g_2^E, c = g_3^E a^{-s'}.$$

$\mathcal{A}$ can answer all queries $j \neq i$ as in Type I. For query i, $\mathcal{A}$ computes $s_i = s' - m_i||r_i$, where $r_i \in_R [0, 2^{l_r})$, and computes

$$v_i = (a^{m_i||r_i + s_i - s'} g_2^{s_i E} g_3^E)^{(e_i^t)^{-1}} = (g_2^{s_i E} g_3^E)^{(e_i^t)^{-1}} = (a^{m_i||r_i} b^{s_i} c)^{(e_i^t)^{-1}}.$$

$\mathcal{A}$ gives $\mathcal{F}$ the signature (v_i, e_i, s_i, r_i) for message m_i, which is statistically indistinguishable from the signature produced by the signature scheme.

Consider now that the forger's signature is (v, e_i, s_i, r) on message m. We have

$$(v/v_i)^{e_i^t} = a^{m||r-m_i||r_i} = g_1^{(m||r-m_i||r_i)E/e_i^t}.$$

Since e_i^t divides E/e_i^t with a negligible probability by Lemma 3.2.2, $|m||r-m_i||r_i| < e_i^t$, and $m_i \neq m$, e_i^t divides $(m||r-m_i||r_i)E/e_i^t$ with a negligible probability, and so with overwhelming probability $GCD(e_i^t, (m||r-m_i||r_i)E/e_i^t) < e_i^t$. Therefore, $\mathcal{A}$ outputs a valid solution to this instance of the multiple generator version of the strong RSA subgroup problem with overwhelming probability. However, this is infeasible under the strong RSA subgroup assumption with multiple generators. We conclude that the $\mathcal{F}$ cannot produce a Type II forgery with non-negligible probability.

Type III: For some $1 \leq i \leq w, e = e_i, s \neq s_i$. $\mathcal{A}$ guesses the forger $\mathcal{F}$ will make the forgery by reusing e_i. $\mathcal{A}$ prepares E as in Type I, and picks at random $s_i \in_R [1, e_i^t)$. Then set up

$$a = g_1^E, b = g_2^{E/e_i^t}, c = g_3^E b^{-s_i}.$$

$\mathcal{A}$ can answer all queries $j \neq i$ as in Type I. For query i, $\mathcal{A}$ computes $v_i = g_1^{(m_i||r_i)E/e_i^t} g_3^{E/e_i^t}$ such that (v_i, e_i, s_i, r_i) is also a valid signature.

Consider now that the forgery $\mathcal{F}$ outputs a new signature (v, e_i, s, r) on message m. We have

$$(v/v_i)^{e_i^t} = a^{m||r-m_i||r_i} b^{s-s_i} = g_1^{(m||r-m_i||r_i)E} g_2^{(s-s_i)E/e_i^t}.$$

Since e_i^t divides E/e_i^t with a negligible probability by Lemma 3.2.2, and $|s-s_i| < e_i^t$, e_i^t divides $(s-s_i)E/e_i^t$ with a negligible probability, and so with overwhelming probability $GCD(e_i^t, (s-s_i)E/e_i^t) < e_i^t$. Therefore, $\mathcal{A}$ outputs a valid solution to this instance of the multiple generator version of the strong RSA subgroup problem with overwhelming probability. However, this is infeasible under the strong RSA subgroup assumption with multiple generators. We conclude that the $\mathcal{F}$ cannot produce a Type III forgery with non-negligible probability. □

4.5 Summary

In this chapter, the new online/offline signing methods are extended and a direct online/offline signature scheme that is secure in the standard model under the strong RSA assumption is proposed.

To pursue solutions for targeted applications presented in Chapter 1, two signature constructions are further devised : the SGE scheme targeted for embedded device authentication, and the SGS scheme targeted for bursty server authentication. The first construction focuses on reducing the offline computation cost while the second construction focuses on reducing the online computation cost. Both the SGE and SGS schemes are proved secure in the standard model under the strong RSA subgroup assumption.

This completes the family of new online/offline signature schemes. While it is not difficult to count the operations required by each of these schemes, the various operations have many complex and interacting parts, so the concrete practical performance gains are not clear from this analysis. To determine if the new designs provide the expected practical improvements, a series of experiments have been conducted comparing the new schemes with each other and with the state-of-the-art schemes in this area. These results are introduced in the next

chapter.

CHAPTER 5

EXPERIMENTS

In this dissertation, five direct online/offline signature schemes based on the new general techniques are proposed. Two of the proposed five schemes, the SGE scheme and the SGS scheme, are the most efficient techniques for embedded device authentication and bursty server authentication, respectively. To fully understand the performance of the proposed schemes, a series of experiments have been conducted comparing the proposed schemes with each other and with other state-of-the-art schemes which have an online/offline signing mechanism, as well as traditional signature schemes in order to appreciate the gains of online/offline signing. For multiprecision arithmetic and standard cryptographic operations, RSARef library is used. The experiments are conducted in two different environments: a desktop computer and AVR studio, a simulation platform for an 8-bit processor.

In this chapter, after the performance of each scheme in this dissertation being analyzed, the experimental environments used are described. Then three different sets of experimental results are shown: The first one is a performance comparison between the traditional signature schemes and their converted constructions using the Shamir-Tauman method in order to verify the effect of online/offline signing and to gain a concrete sense of times involved for signature generation in the online phase; The second one is the comparison of offline performance among all schemes in order to explore which scheme is best suited for embedded device authentication scenario; The third one focuses on the online performance of all schemes in order to identify the best scheme for the bursty server authentication scenario.

5.1 Performance Analysis

First analyze the performance of the schemes in this dissertation. It can be seen that majority of computation in these schemes are due to modular exponentiations. Thus, the performance is mainly determined by the cost of modular exponentiations in these schemes. The technique for general modular exponentiation is called the "repeated square-and-multiply" algorithm [30]. There are several variants for this algorithm, and one of them, called "Right-to-left binary exponentiation", is introduced as follows.

Definition 5.1.1 (Right-to-Left Binary Exponentiation) *For an element $g \in G$ and integer $r \geq 1$, g^r is obtained through the following procedure.*

1. $A \leftarrow 1$, $S \leftarrow g$.
2. *While* $r \neq 0$
   (a) *if* r *is odd, then* $A \leftarrow A \times S$.
   (b) $r \leftarrow \lfloor r/2 \rfloor$
   (c) *if* $r \neq 0$, *then* $S \leftarrow S \times S$
3. *Return (A)*

Let $k+1$ be the bit length of the binary representation of r, and let $wk(r)$ be the number of $1's$ in this representation. The algorithm requires k squarings and $wk(r)-1$ multiplications. If r is randomly selected in the range $[0, n)$, then $\lfloor lg(n) \rfloor$ squarings and $\frac{1}{2}(\lfloor lg(n) \rfloor + 1)$ multiplications can be expected. Since the cost of a multiplication can be

twice as that of a squaring [30], two squarings are treated as one multiplication, so the expected cost is about $\lfloor lg(n) \rfloor$ multiplications.

To take an example from the proposed signature schemes, if the security parameter $k = 512$ in the SQ scheme, the order of QR_n is $p'q'$, which is roughly 1022 bits long in binary representation. Therefore, $lg(|QR_n|)$ would be about 1022, so the cost for the offline phase of the signing algorithm would be around 1022 modular multiplications.

The algorithms in the schemes in this dissertation only require several modular exponentiations and additional computation if needed. The bit length of exponents in these schemes is polynomial in the security parameter k, so all these schemes can be efficiently completed in time that is polynomial in k. To simplify the analysis, the efficiency of a scheme can be estimated by the total bit length of exponents in this scheme.

5.2 Overview of the Experiments

This section reviews the schemes implemented in the experiments, experimental environments used, and the three sets of experiments conducted in the experiments.

5.2.1 The Schemes in the Experiments

For comparison with the new schemes, state-of-the-art schemes which provide online/offline signing mechanisms, are implemented, including constructions based on Shamir and Tauman's "Hash-Sign-Switch" paradigm, and constructions with direct online/offline schemes such as the Schnorr scheme. The primary interest is in schemes proved secure in the standard model, and which are designed for the two sample scenarios. Other widely used techniques, which can only be proved secure in the random oracle model, are included, such as the RSA

scheme and the Schnorr scheme, since they are popular. Following are the schemes that are used in the experiments.

5.2.1.1 The RSA Scheme

The RSA scheme is the most widely used signature scheme in practice, which is introduced in Chapter 1. The RSA scheme operates in the group Z_n^*, having a long exponentiation for the signing algorithm, but a very efficient exponentiation with a short exponent for the verification algorithm (a commonly used exponent is 65537, which is 0x10001 in hexadecimal form, so this exponentiation only requires 16 squarings and one multiplication).

5.2.1.2 The CS Scheme

The CS scheme, proposed by Cramer and Shoup in 2000, was the first practical signature scheme to be proved secure in the standard model [12]. The scheme works in QR_n for n being a special RSA modulus. The CS scheme is described here for completeness.

Key Generation. k is the security parameter, and l is an additional parameter, which is polynomial in k, where $l+1 < k$. On input 1^k, pick two k-bit safe primes p, q and let $n = pq$. Select $a, b \in QR_n$. Select a random $(l+1)$-bit prime e'. Select a collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$. The public key is (n, a, b, e', h), and the private key is (p, q).

Signing Algorithm. To sign a message m, choose a random $(l+1)$-bit prime $e \neq e'$ and a random $v' \in QR_n$. The equation

$$v^e = ab^{h(x')} \mod n$$

is solved for v, where x' satisfies the equation

$$v'^{e'} = x' b^{h(m)} \mod n.$$

The signature is (e, v, v').

Verification Algorithm. To verify that (e, v, v') is a signature on message m, check that e is an odd $(l+1)$-bit number different from e'. Next, compute $x' = v'^{e'} b^{-h(m)} \mod n$ and check whether $a \equiv v^e b^{-h(x')} \mod n$.

The CS scheme needs about three short (l-bit) exponentiations, one long ($2k$-bit) exponentiations, generation of an $l+1$-bit prime number, and computing the multiplicative inverse of e for the signing algorithm. Thus, the CS scheme is several times slower than the RSA scheme.

5.2.1.3 The CL Scheme

The CL scheme was proposed by Camenisch and Lysyanskaya in 2002. The SQ scheme in Chapter 4 can be considered as the online/offline extension of the CL scheme.

Key Generation. k is the security parameter. l is a parameter that controls the statistical closeness of distributions, and should be at least polynomial in k. On input 1^k, pick two safe k-bit safe primes p, q and let $n = pq$. Select $a, b, c \in_R QR_n$. The public key is (n, a, b, c), the private key is (p, q).

Signing Algorithm. The message space is $[0, 2^{l_m})$. Given a message m, pick a random prime number e with length $l_e \geq l_m + 2$, and a random number s of length $l_s = l_n + l_m + l$,

and solve for v in the equation

$$v^e = a^m b^s c \mod n.$$

(v, e, s) is the signature on m.

Verification Algorithm. To verify that (v, e, s) is a signature on message m, check that e's length is l_e, and $v^e \equiv a^m b^s c \mod n$.

The CL scheme requires one short (l_m-bit) and two long (l_n-bit and l_s-bit) exponentiations, generation of l_e-bit prime number, and computing the multiplicative inverse of e for the signing algorithm.

5.2.1.4 The Schnorr Scheme

The Schnorr scheme was designed for smart cards with limited computing capabilities. This scheme provides a direct online/offline signing mechanism.

Key Generation. k is the security parameter. l is a parameter, and should be at least polynomial in k. On input 1^k, pick one l-bit prime number q, and one k-bit prime number p such that $p = q \times t + 1$ for a random t. Pick a random b whose order is q, i.e., $b^q = 1 \mod p$. Pick $\alpha \in_R (0, q)$, let $a = b^\alpha \mod p$. Pick a collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$. The public key is (p, q, a, b, h), the private key is α.

Signing Algorithm. The signing procedure includes two phases.

Offline Phase: The signer picks a random $\beta \in_R [0, q)$, computes

$$v = b^\beta \mod p.$$

ONLINE PHASE: When a message m appears, the signer computes

$$r = h(m||v),\ w = r \times \alpha + \beta \mod q.$$

The signature is (r, w).

Verification Algorithm. To verify that (r, w) is a signature on message m, compute $v' = b^w a^{-r} \mod p$, and check that whether $h(m||v') = r$.

The Schnorr scheme is a very efficient online/offline scheme, with one short (l-bit) exponentiation in the offline phase, and one hash calculation, one modular multiplication, and one modular addition in the online phase. However, the Schnorr scheme can only be proved secure in the random oracle model.

5.2.1.5 The SQ Scheme

This is the basic signature scheme which is secure in the standard model. It needs one long exponentiation and generation of an l_e-bit prime in the offline phase, one multiplication and addition in the online phase.

5.2.1.6 The SGE Scheme

This scheme is specialized for embedded device authentication scenario. In the offline phase, it requires one short exponentiation, a hash calculation and several multiplications. In the online phase, it needs three multiplications with small numbers and three additions.

5.2.1.7 The SGS Scheme

This scheme is specialized for bursty server authentication. In the offline phase, it requires one short exponentiation, generation of a medium size (e.g., 88-bit) prime number, and several multiplications. In the online phase, it only requires one addition and one modular reduction with a short modulus.

5.2.2 Experimental Environments

The experiments are conducted in two different environments. The majority of the experiments have been performed on a desktop class computer. The specifics of this environment are as follows:

- CPU: Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz;
- Memory: 2031612 KB;
- Operating System: Linux 2.6.23.15-80.fc7;
- Compiler: GCC Version 4.1.2.

To test the behavior of the schemes on embedded devices with limited computing capabilities, experiments are also conducted using Atmel's AVR Studio 4.0 (Version 4.13 Service Pack 2) [2], a simulation platform for an 8-bit processor that is popular for embedded systems. The experiments in AVR Studio simulate the following system:

- CPU: atmega2560 16MHz;
- Memory: 255 KB (Program Memory), 8 KB (Data Memory);

- Compiler: WinAVR-20071221.

The experiments in AVR Studio focus on answering the question of whether the proposed schemes operate in an acceptable amount of time on such a limited processor, so the tests measure the time for the online phase and offline phase of these algorithms when the key size is 1024 bits. While more extensive tests would provide some additional information that could be interesting, the extreme slowness of simulation in AVR Studio makes this infeasible. Each second of simulation time takes about 3.6 minutes of real time, meaning that a single trial of the complete signing procedure in RSA-ST, which takes 200 seconds of simulated time, takes almost 12 hours of real time. Therefore, the experiments in AVR Studio are limited to only the more important and interesting questions.

For multiprecision arithmetic and standard cryptographic operations, RSARef [37] is used, which is a cryptographic toolkit designed by RSA Laboratories, a division of RSA Data Security, Inc. RSARef means "RSA reference" and serves as a portable, educational, reference implementation, which is available for free.

5.2.3 Experiments Performed

The experiments performed focus on online/offline signing for each scheme in this dissertation. Three sets of experiments have been carried out as described below.

- The first set of experiments is a performance comparison between the traditional signature schemes and their converted constructions using the Shamir-Tauman method. The purpose for this set of experiments is to verify the effect of online/offline signing, to gain a concrete sense of times involved for signature generation in the online phase.

- The second set of experiments focuses on the offline performance among all schemes. Since the embedded device scenario depends on the offline phase being efficient (as well as the online phase), these experiments explore which scheme is best suited for embedded device authentication.

- The last set of experiments focuses on the online performance of all schemes, in order to identify the best scheme for the bursty server authentication scenario.

For each set of experiments, key sizes from 512 bits to 2048 bits are tested in order to obtain performance data under different key sizes. In addition, some tests on the verification phase are also run to provide a complete picture for each scheme.

5.3 Experimental Results

The experimental results are presented in this section. In the experiments, five different sets of parameters are generated for each signature scheme, and for each set of parameters and each scheme, the signature algorithm is run for 100 different messages, then the time for each phase such as offline phase, online phase and verification phase, is averaged. Since the exponents in each scheme are randomly chosen, the efficiency of the scheme is determined by the total bit length of exponents in this scheme as described in Section 5.1, so the averaged value should be similar for different test times. Averaging over different numbers of signature tests are tried, and the average of 100 gave consistent, reliable results. Therefore, 100 is a reasonable test times for each set of parameters of each scheme in the experiments. The data in the following tables are averaged value for time in seconds.

5.3.1 Testing the Shamir-Tauman Method

This set of the experiments is designed to explore the efficiency of Shamir and Tauman's "Hash-Sign-Switch" paradigm, and to get a feel for how much online/offline techniques can improve efficiency on desktop class systems. RSA, CS, CL, and their corresponding online/offline versions based on the Shamir-Tauman method, which are called RSA-ST, CS-ST, and CL-ST, are implemented. Table 5.1 gives the experimental results for average online signing time with different key sizes. Figure 5.1 shows these results on a graph, and while the scale required to show all six results does not allow any difference to be seen among the three ST-based techniques, it does show the dramatic improvement over the non-online/offline versions. For example, when the key size is 1024 bits, which is a typical key size used in practice, the online phase of RSA-ST is over 7538 times faster than using RSA in a traditional non-online/offline manner. Figure 5.2 shows that all RSA-ST, CS-ST, and CL-ST line up practically on top of each other, which reflects the fact that the online phase of the ST-based techniques depends on the trapdoor hash used, but not on the underlying traditional signature scheme. Thus, their online signing times are almost the same.

Key Size	RSA	CS	CL	RSA-ST	CS-ST	CL-ST
512	0.0064	0.0252	0.0212	0.00000312	0.00000325	0.00000304
768	0.0178	0.0542	0.0512	0.00000375	0.00000384	0.00000373
1024	0.0340	0.1082	0.1078	0.00000451	0.00000449	0.00000451
1280	0.0742	0.1948	0.1930	0.00000520	0.00000551	0.00000545
1536	0.1230	0.3212	0.3208	0.00000588	0.00000602	0.00000597
1792	0.1944	0.4741	0.4781	0.00000681	0.00000669	0.00000704
2048	0.2826	0.7073	0.7167	0.00000742	0.00000770	0.00000763

Table 5.1: Online Performance Comparison of Traditional Signature Schemes and their ST-based Signature Schemes (times in seconds on a Pentium Core 2 Duo E6750)

While 34 ms for RSA is acceptable for a single request in most situations, when there is a

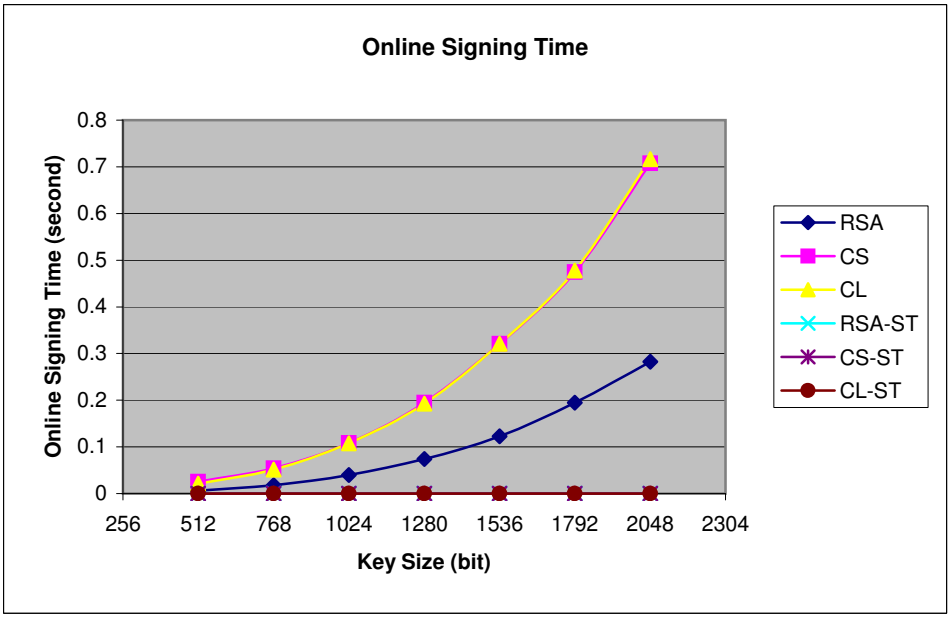

Figure 5.1: Online Performance Comparison of Traditional Signature Schemes and their ST-based Signature Schemes

Figure 5.2: Online Performance of the ST-based Signature Schemes

heavier load of requests this time becomes quite significant, and the 7538 times improvement from using the Shamir-Tauman method is very dramatic. So, this set of experiments demonstrates the Shamir-Tauman techniques is an effective technique to achieve a online/offline signing.

5.3.2 Offline Performance

As described in Chapter 1, offline performance is important in the embedded device authentication scenario. As seen in the previous section, the Shamir-Tauman technique can dramatically improve the online signing speed, but as can be seen from the algorithm definition this is paid for with additional computation in the offline phase. In this section, experiments are run to determine how much this affects times in practice, and compare to the proposed direct online/offline schemes to judge whether improvements are significant.

Table 5.2 gives the experimental results for offline performance of all online/offine signature schemes. Comparing Table 5.1 and Table 5.2, it can be seen that the Shamir-Tauman method adds significant cost to the original signature scheme. For example, the signing time for the RSA scheme approximately doubles, increasing from 34ms to 79ms with 1024-bit keys. This additional overhead is significant considering the limited computing capabilities of embedded devices. By comparison, the direct online/offline schemes in the rightmost 4 columns of Table 5.2 show very significant improvements in running time, so in the following the attention is restricted to just the direct online/offline schemes.

Figure 5.3 compares the four direct online/offline schemes that are tested on a graph. The Schnorr, SGE and SGS's performance stand almost at the same level, with the Schnorr scheme being slightly faster than SGE and SGS.

Key Size	RSA-ST	CS-ST	CL-ST	Schnorr	SQ	SGE	SGS
512	0.0128	0.0286	0.0266	0.0008	0.0092	0.0014	0.0026
768	0.0360	0.0736	0.0726	0.0029	0.0198	0.0040	0.0054
1024	0.0792	0.1498	0.1520	0.0072	0.0396	0.0096	0.0104
1280	0.1458	0.2670	0.2722	0.0145	0.0734	0.0176	0.0182
1536	0.2398	0.4525	0.4563	0.0246	0.1182	0.0302	0.0312
1792	0.3777	0.6845	0.6853	0.0409	0.1848	0.0458	0.0478
2048	0.5525	0.9994	0.9926	0.0608	0.2778	0.0684	0.0688

Table 5.2: Offline Performance of Online/Offline Signature Schemes (times in seconds on a Pentium Core 2 Duo E6750)

While the initial experiments were in a standard PC environment, the embedded device authentication scenario demands that performance be tested in a typical embedded device environment. For this purpose, the AVR Studio simulator is used, as described earlier, and Table 5.3 shows the simulation results in the AVR Studio environment.

	RSA	RSA-ST	Schnorr	SGE	SGS
Offline	N/A	212.24	21.69	24.11	26.69
Online	105.99	0.0138	0.0191	0.0149	0.0077

Table 5.3: Online/Offline Performance (times in seconds on AVR Studio)

Using typical security parameters and simulating an Atmega2560 processor, the Schnorr scheme is about 10% more efficient than the SGE scheme for the offline signing, but the security guarantees of the SGE scheme are higher since it does not require the random oracle model to reason about security. The 10% is potentially significant, but SGE can have stronger security guarantees. Since SGE and SGS have the same basic security assumptions, the one with best performance (SGE) should be used. By comparison, traditional RSA and RSA with the Shamir-Tauman online/offline extension are much less suitable for the embedded device scenario due to a much larger signing time.

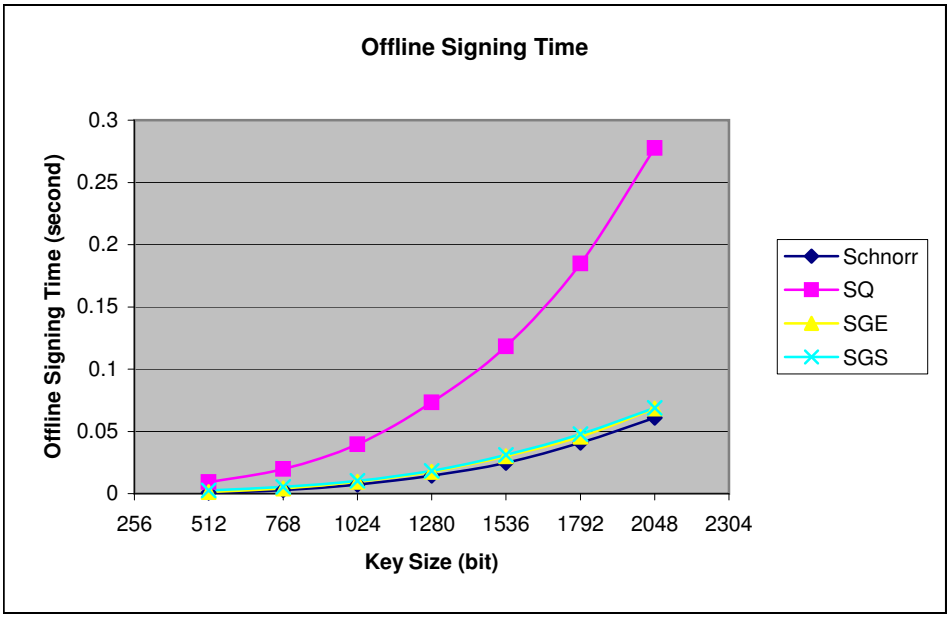

Figure 5.3: Offline Performance of Direct Online/Offline Signature Schemes

In the scenario for embedded device authentication, since these devices have limited computing capabilities and most of them are powered by batteries, more computation implies more energy use, and shorter life cycle. Therefore, idle-time computation for the offline phase is not free as it is for servers, it is important to reduce offline computation cost (and hence energy use) as much as possible for embedded devices. The new direct online/offline techniques avoid overhead associated with the offline phase of the Shamir-Tauman technique, and use computation over the small subgroup G of Z_n^*, which results in much shorter modular exponentiations. The experiments show that this makes a concrete and noticeable difference in practice: the offline phase of the SGE technique is roughly 8 times faster than RSA-ST for a typical key size. Thus, the experimental results show that the SGE and SGS schemes are more suitable for the embedded device scenario.

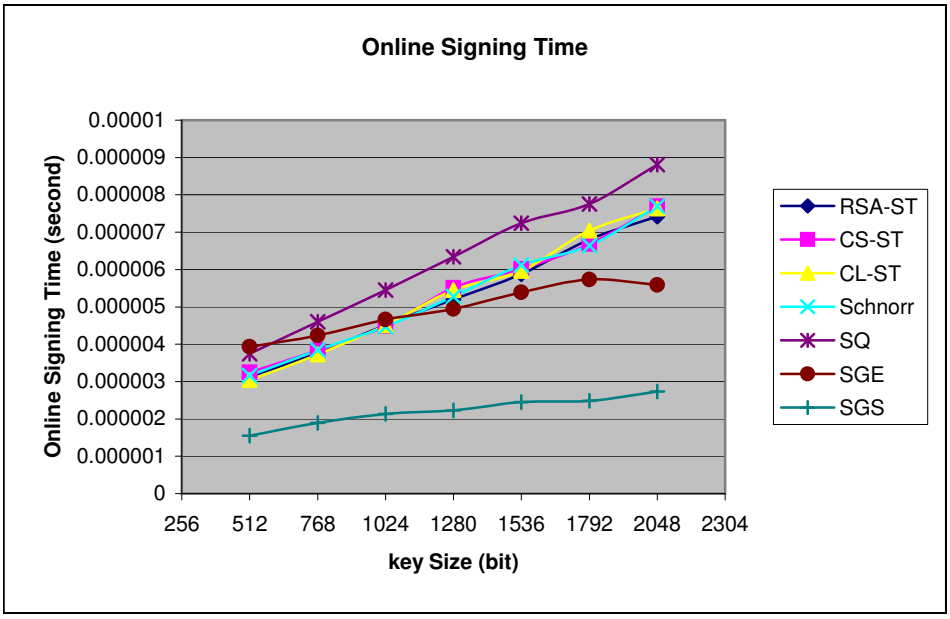

Figure 5.4: Online Performance of Online/Offline Signature Schemes

5.3.3 Online Performance

Online performance is key to the bursty server authentication scenario, and in this section the online performance of the various online/offline signature schemes is carefully examined. Table 5.4 and Figure 5.4 give the experimental results on the online performance of all schemes.

Key Size	RSA-ST	CS-ST	CL-ST	Schnorr	SQ	SGE	SGS
512	3.1	3.2	3.0	3.2	3.7	3.9	1.6
768	3.8	3.8	3.7	3.8	4.6	4.2	1.9
1024	4.5	4.5	4.5	4.5	5.5	4.7	2.1
1280	5.2	5.5	5.4	5.3	6.3	4.9	2.2
1536	5.9	6.0	6.0	6.1	7.2	5.4	2.4
1792	6.8	6.7	7.0	6.6	7.8	5.7	2.5
2048	7.4	7.7	7.6	7.7	8.8	5.6	2.7

Table 5.4: Online Performance of Online/Offline Signature Schemes (times in microseconds on a Pentium Core 2 Duo E6750)

To get a better idea of how these schemes perform in the bursty server scenario, we

look at these performance results in terms of throughput, i.e., how many signatures can be produced in one second. Note that if we were to consider the long term throughput, we would have to factor in the time for the offline phase as well. For example, if a server repeats busy/slow cycles on a daily basis, and in one day can finish offline computation for at most one million signatures, then no matter how fast the online phase is, this will be the maximum throughput for this server in a day. However, note that the online/offline model also provides an excellent means for supporting throughput that can be scaled up by adding additional machines: by having a cluster of machines working on offline phase precomputation and making the results available to the front-end machine, throughput can be increased almost to the full online signature throughput. Therefore, the main limiting factor: the online phase throughput, is considered in this dissertation.

Data from RSA, RSA-ST, Schnorr, SGE, and SGS are selected to measure bursty authentication throughput. Note that when the Shamir-Tauman technique is used, the online phase is independent of the underlying signature scheme, so the results for RSA-ST can be used for CS-ST and CL-ST as well.

Table 5.5 and Figure 5.5 show the performance data in terms of bursty authentication throughput. Clearly, the SGS scheme outperforms all other schemes by a substantial amount. For example, for a 1024-bit key,the SGS scheme generates over twice as many signatures as any other scheme: 2.188 times signatures as many as the SGE scheme, 2.098 times as many as the Schnorr scheme, and 2.12 times as many as the RSA-ST scheme.

Key Size	RSA	RSA-ST	Schnorr	SGE	SGS
512	156	320,153	315,905	254,362	643,583
768	56	265,999	259,780	236,440	528,066
1024	25	221,273	223,548	214,348	469,109
1280	13	192,263	189,422	202,212	449,296
1536	8	170,151	163,476	185,556	408,563
1792	5	146,907	150,534	174,547	402,965
2048	3	134,718	129,890	179,108	366,085

Table 5.5: Bursty Authentication Throughput (online signatures per second)

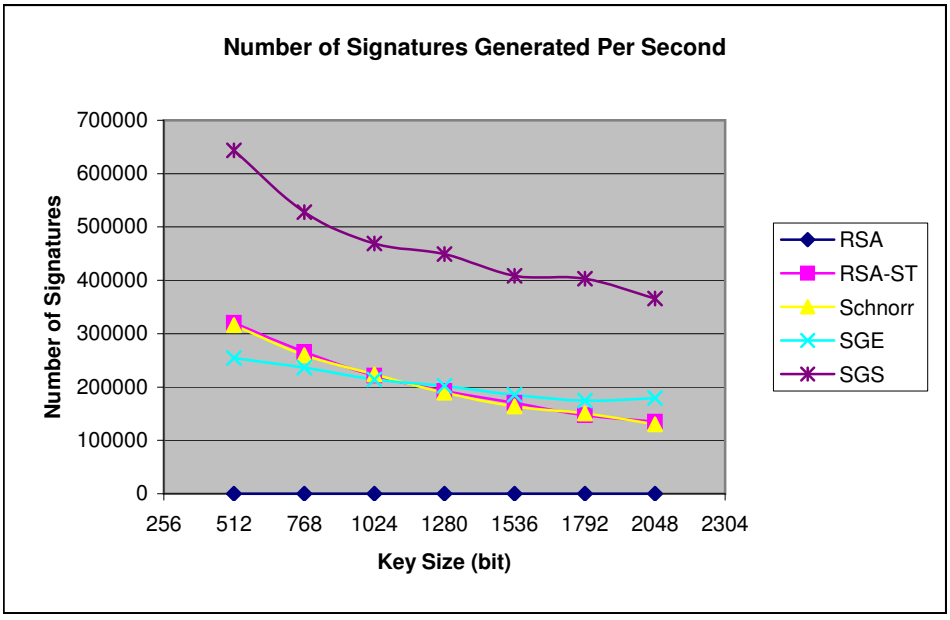

Figure 5.5: Bursty Authentication Throughput

5.3.4 Verification Performance

In this dissertation, signing performance for digital signature schemes, both offline and online phases, as appropriate for the application scenarios, are mainly considered. Experiments are also conducted to measure the verification phase, to provide a complete picture for each scheme.

Table 5.6 and Figure 5.6 give these experimental results. The verification of each scheme

Key Size	RSA-ST	Schnorr	SQ	SGE	SGS
512	0.0072	0.0032	0.0132	0.0194	0.0160
768	0.0190	0.0077	0.0312	0.0412	0.0338
1024	0.0410	0.0156	0.0618	0.0716	0.0620
1280	0.0746	0.0271	0.1080	0.1124	0.1054
1536	0.1226	0.0415	0.1718	0.1646	0.1586
1792	0.1906	0.0646	0.2538	0.2270	0.2144
2048	0.2796	0.0909	0.3701	0.3012	0.2872

Table 5.6: Verification Performance (times in seconds on a Pentium Core 2 Duo E6750)

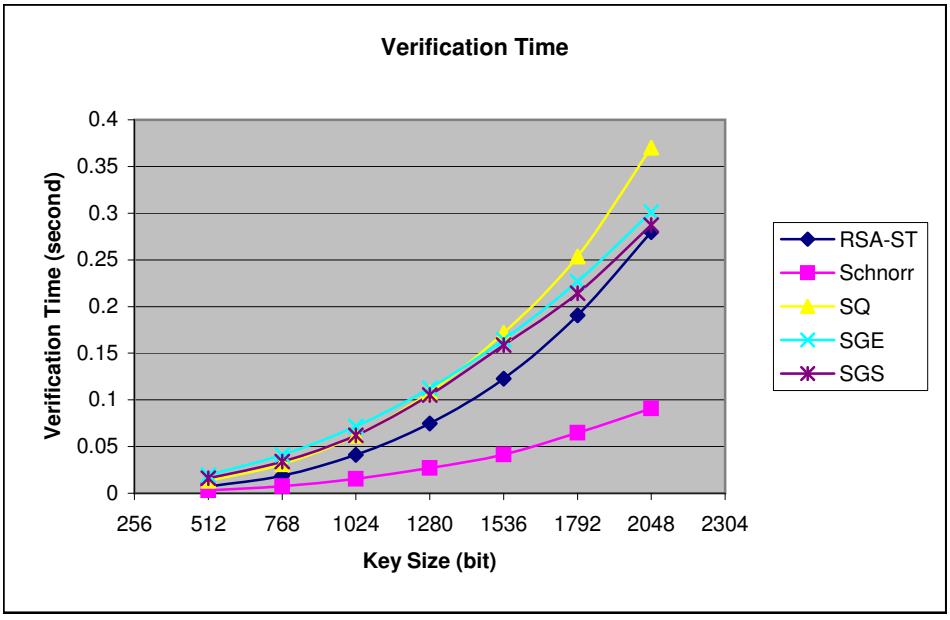

Figure 5.6: Verification Performance

is less than half second, which is very fast considering verifications are only needed and performed occasionally on a computer rather than a embedded device in the sample scenarios. It can be seen that signatures from the Schnorr scheme can be verified the fastest of all schemes, and the SGE and SGS schemes are a little slower than the Schnorr scheme and the RSA-ST scheme for verification performance. However, this little slowness in verification is trivial for the sample scenarios, and the SGE and SGS schemes are still the best schemes for embedded device authentication scenario and bursty server authentication scenario, respectively.

5.4 Summary

In this chapter, three different sets of experimental results have been shown. The first set of experiments between the traditional signature schemes and their ST-based signature schemes confirmed that online/offline signature schemes are very important in the sample scenarios; The second set of comparison experiments of offline performance among all schemes showed that the SGE scheme is more suitable for embedded device authentication scenario; The last set of comparison experiments of online performance among all schemes demonstrated that the SGS scheme is the best scheme for the bursty server authentication scenario.

In the next chapter, the research results will be summarized, and conclusions and future research for the dissertation will be given.

CHAPTER 6

CONCLUSION

In this dissertation, new general techniques for online/offline signing were developed, those were applied in a variety of ways for creating online/offline signature schemes, and five different online/offline signature schemes that are proved secure under a variety of models and assumptions were proposed. The results of the new techniques include two direct online/offline schemes that have the best offline or best online performance of any currently known technique. These two schemes are particularly useful in the two scenarios defined in Chapter 1: bursty server authentication and embedded device authentication. Through a series of experiments we have gained a concrete understanding of the performance of the new techniques, in relation to each other and to other state-of-the-art techniques. The new techniques are able to provide online signature performance that produces over twice as many signatures per second as the next best technique, and have the best offline performance of any scheme that has been proven secure in the standard model.

After the background and foundation for the research work being introduced in Chapters 1 and 2, the core design technique used in this dissertation, the two-exponent version of the flexible RSA problem, was introduced, which provides the flexibility of performing some computations before the message to be signed is known. As a direct application of this technique, the new direct online/offline signature scheme, the RQ scheme, was proposed. Although the RQ scheme is not a fully optimized construction, it is the foundation for the further improvements. Using computation over a small subgroup of Z_n^*, another simple and

efficient direct online/offline signature construction, the RG scheme, was further devised. The RQ and RG schemes are proved secure under adaptive chosen message attack with the strong RSA assumption and the strong RSA subgroup assumption, respectively. However, both security proofs rely on the random oracle model.

Next, the underlying reason for the random oracle being necessary in the proofs for the RQ and RG schemes was analyzed. As a result, a direct online/offline signature scheme, the SQ scheme, was devised, which is secure in the standard model under the the strong RSA assumption. Based on the SQ scheme, first computation over a small subgroup of Z_n^* was used to reduce overall computation overhead. Looking specifically at the offline phase of the algorithm, in the next improvement a type of hash function called a division intractable hash function was used to further reduce the offline cost, and the SGE scheme was obtained, which targets the scenario of embedded device authentication. The SGE scheme has the best offline performance of any known signature scheme that is provably secure in the standard model. In the last derivation, Shamir and Tauman's trapdoor hash function was adapted into the new direct online/offline method, and the most efficient construction in the online phase, the SGS scheme, was obtained, which targets the scenario of bursty server authentication. The SGS scheme has the best online performance of any known scheme, even compared against those that are secure only in the random oracle model. Both the SGE and SGS schemes are proved secure in the standard model under the strong RSA subgroup assumption.

Finally, to determine if the new direct online/offline designs provide the expected practical improvements, a series of experiments have been conducted comparing the new schemes with each other and with other state-of-the-art schemes in this area, both on a desktop class computer, and under AVR studio, a simulation platform for an 8-bit processor that is

popular for embedded systems. The experimental results also show that the SGE scheme and SGS scheme are the most efficient techniques for embedded device authentication and bursty server authentication, respectively.

6.1 Future Research

In this dissertation, five direct online/offline signature schemes based on the new general techniques has been successfully devised. Two of the proposed five schemes, the SGE scheme and the SGS scheme, are the most efficient online/offline techniques of any known scheme in the standard model. Both of them operate on a small subgroup G of Z_n^* in order to reduce overall computation overhead, and so they are proved secure under the strong RSA subgroup assumption, which is stronger than the strong RSA assumption.

The strong RSA subgroup assumption is only a few years old, so is relatively new by cryptographic standards. Therefore, one important direction for future work is to pursue a better understanding of the relation between the strong RSA assumption and the strong RSA subgroup assumption. A stronger foundation in this area could allow us more insight into the selection of parameters for subgroup-based schemes, allowing us to optimize various aspects more carefully.

The RQ and RG schemes in Chapter 3 can be proved secure in the random oracle model. The question is: can we determine if random oracles are really required in their security proofs? Therefore, another important direction for future work is to have a deeper understanding of the random oracle model, which could allow us to better understand the requirements on the hash function, the real difference between a signature scheme in the random oracle model and a signature schemes in the standard model, and the problems with

random oracles.

BIBLIOGRAPHY

[1] 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.

[2] Atmel. AVR studio 4.0 (version 4.13 service pack 2), 2007. http://www.atmel.com/products/AVR/.

[3] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — Eurocrypt'97*, pages 480–494, 1997.

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

[5] 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.

[6] M. Bellare and P. Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In *Advances in Cryptology — Eurocrypt'96*, pages 399–416, 1996.

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

[8] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — Crypto'02, LNCS 2442*, pages 61–76, 2002.

[9] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Third Conference on Security in Communication Networks (SCN'02)*, pages 268–289, 2002.

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

[11] R. Cramer and I. Damgard. New generation of secure and practical RSA-based signatures. In *Advances in Cryptology — Crypto'96*, pages 173–185, 1996.

[12] R. Cramer and V. Shoup. Signatures schemes based on the strong RSA assumption. In *ACM Transaction on Information and System Security*, pages 161–185, 2000.

[13] I. Damgard. Collision free hash functions and public key signature schemes. In *Advances in Cryptography – Eurocrypt'87*, pages 203–216, 1987.

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

[15] C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. *J. Cryptology*, 11(3):187–208, 1988.

[16] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — Crypto'84*, pages 10–18, 1984.

[17] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *Advances in Cryptology — Crypto'89*, pages 263–275, 1990.

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

[19] M. Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In *International Workshop on Practice and Theory in Public Key Cryptography (PKC 2003)*, pages 116–129, 2003.

[20] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — Crypto'97*, pages 16–30, 1997.

[21] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology — Eurocrypt'99*, pages 123–139, 1999.

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

[23] S. Goldwasser and Y. T. Kalai. On the (in)security of Fiat-Shamir paradigm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science — FOCS'03*, pages 102– 114, 2003.

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

[25] J. Groth. Cryptography in subgroups of Z_n^*. In *Theory of Cryptography Conference (TCC 2005)*, pages 50–65, 2005.

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

[27] H. Krawczyk and T. Rabin. Chameleon signatures. In *Symposium on Network and Distributed Systems Security – NDSS'00*, pages 143–154, 2000.

[28] L. Lamport. Constructing digital signatures from a one-way function, Oct. 1979.

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

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

[31] D. Naor, A. Schenhav, and A. Wool. One-time signatures revisited: Practical fast signatures using fractial merkel tree traversal. In *IEEE 24th Convention of Electrical and Electronics Engineers i Israle*, pages 255–259, 2006.

[32] NIST. Data encryption standard (des), 1999. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[33] NIST. Secure hash standard, 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

[34] M. O. Rabin. Digitalized signatures. In *Foundations of Secure Computation*, pages 155–168, 1978.

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

[36] RSA. The RSA factoring challenge.

[37] RSA Laboratories. RSAREF(TM): A cryptographic toolkit for privacy-enhanced mail, 1993. http://plan9.bell-labs.com/sources/contrib/btdn/src/pgp/rsaref/.

[38] C. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[39] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *Advances in Cryptology — Crypto'01*, pages 355–367, 2001.

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

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

[42] P. Yu and S. R. Tate. An online/offline signature scheme based on the strong RSA assumption. In *21st International Conference on Advanced Information Networking and Applications Workshops – 3rd IEEE International Symposium on Security in Networks and Distributed Systems (SSNDS)*, pages 601–606, 2007.

[43] P. Yu and S. R. Tate. Online/offline signature schemes for devices with limited computing capabilities. In *RSA Conference 2008, Cryptographers' Track (CT-RSA)*, pages 301–317, 2008.

[44] H. Zhu. New digital signature scheme attaining immunity to adaptive chosen-message attack. *Chinese Journal of Electronic*, 10(4):484–486, 2001.

[45] H. Zhu. A formal proof of Zhu's signature scheme, 2003. http://eprint.iacr.org/.