Digital Signcryption

by

Clayton D. Smith

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2005

©Clayton D. Smith 2005

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Signcryption is a new cryptographic primitive which simultaneously provides both confidentiality and authenticity. Previously, these two goals had been considered separately, with encryption schemes providing confidentiality and signature schemes providing authenticity. In cases where both were required, the encryption and signature operations were simply sequentially composed. In 1997, Zheng demonstrated that by combining both goals into a single primitive, it is possible to achieve significant savings both in computational and communication overhead. Since then, a wide variety of signcryption schemes have been proposed.

In this thesis, we present a number of the proposed signcryption schemes in terms of a common framework. For the most part, the material has been previously presented in various research papers, but some previously omitted proofs have been filled in here. We begin by giving a formal definition of the signcryption primitive, complete with a security model. Then we look at some of the various proposed signcryption schemes, and consider their relative advantages and disadvantages. Finally, we look ahead at what future progress might be made in the field.

Acknowledgements

I would like to thank my supervisor, Edlyn Teske, for proposing the topic of signcryption, as well as providing numerous comments, suggestions, and corrections during the writing of this thesis. I would also like to thank Yevgeniy Dodis for his very prompt response to my questions regarding his paper. Finally, I would like to thank my readers, Alfred Menezes and Doug Stinson, for their valuable comments and corrections.

Contents

List of Tables

Chapter 1

Introduction

1.1 What is Signcryption?

Of the many goals which the study of cryptography sets out to achieve, the most important and widely studied are *confidentiality* and *authenticity*. Traditionally, these two goals have been studied separately. In the case of public-key cryptography, confidentiality is provided by encryption schemes, while authenticity is provided by signature schemes.

In many applications, both confidentiality and authenticity are needed together. Such applications include secure email (S/MIME), secure shell (SSH), and secure web browsing (HTTPS). Until recently, the de facto solution was to use both an encryption scheme and a signature scheme, typically by sequentially composing the encryption and signature operations. This state of affairs changed in 1997, when Zheng [15] proposed using a single cryptographic primitive to achieve both confidentiality and authenticity. He called this primitive *signcryption*.

At first glance, it is not clear why there should be any advantage to lumping both goals into a single primitive. However, Zheng and others have demonstrated, through concrete examples, that signcryption schemes can provide clear benefits over the traditional sequential composition of encryption and signature schemes. For example:

- It is possible to create signcryption schemes which are more efficient, in terms of computational complexity and/or communication overhead, than the sequential composition of signature and encryption schemes. We will see concrete examples of such schemes in Chapters 4 and 5.

- Many signcryption schemes require only a single key pair for each user,

while the traditional approach requires two: one for encrypting and one for signing.

- Some signcryption schemes allow expensive cryptographic operations to be parallelized.
- In its most basic form, the sequential composition of signature and encryption schemes yields only a weakly secure signcryption scheme, as we shall see in Chapter 3. With some slight modifications, it is possible to achieve a much stronger level of security. This example suggests that it is worthwhile to work inside the framework of signcryption whenever both authenticity and confidentiality are required.
- The availability of a signcryption primitive can simplify the design of cryptographic protocols which require both authenticity and confidentiality. For example, Dodis et al. [5] have given a very simple signcryption-based authenticated key-exchange protocol.

It is clear from this list that signcryption is a topic worthy of further study, and that it should be considered whenever designing systems that require both confidentiality and authenticity.

1.2 Outline

We begin in Chapter 2 by considering what properties every signcryption scheme should have, and what additional properties would be desirable. In Chapter 3 we give a formal definition of signcryption, complete with a security model. We also look at several signcryption schemes which can be built from plain encryption and signature schemes. In Chapter 4 we explore a variety of signcryption schemes based on the discrete logarithm problem. In Chapter 5 we do the same for schemes based on trapdoor one-way permutations. In Chapter 6 we compare the various schemes presented in previous chapters, and we conclude in Chapter 7.

1.3 Scope

Since a great deal of research has already been done in the field of signcryption, it is impossible to cover the entire field here. The signcryption schemes which are presented here certainly do not constitute an exhaustive list, but instead are intended to give an overview of the various research directions which have been explored so far.

The material presented here is, for the most part, a unified presentation of existing work. However, some new material is presented, mainly in the form of proofs which were omitted from the original research papers. These proofs are indicated as such where they appear.

A few veins of research are explicitly not considered here. The most notable is identity-based signcryption, which is an adaptation of identity-based encryption to the case of signcryption. Interested readers should begin with the paper of Malone-Lee [7].

Chapter 2

Properties of Signcryption Schemes

In this chapter, we take an informal look at the most important properties of signcryption schemes, and explain their connection to the formal definition of signcryption, which will be presented in Chapter 3.

2.1 Confidentiality and Authenticity

At the very least, we would expect a signcryption scheme to achieve the basic goals of encryption and signature schemes, namely confidentiality and authenticity.

By *confidentiality*, we mean that only the intended recipient of a signcrypted message should be able to read its contents. That is, upon seeing a signcrypted message, an attacker should learn nothing about the original message, other than perhaps its length.

By *authenticity*, we mean that the recipient of a signcrypted message can verify the sender's identity. That is, an attacker should not be able to send a message, claiming to be someone else.

It is these two properties which we require of all signcryption schemes. They will form the security portion of the formal definition of signcryption.

2.2 Non-repudiation and Signature Verifiability

By *non-repudiation*, we mean that the sender of a message cannot later deny having sent the message. That is, the recipient of a message can prove to a third party that the sender indeed sent the message.

Note that signature schemes always provide non-repudiation, since anyone can verify a signature using only the sender's public key. That is, the recipient of a signed message need only forward it to a third party, who can then verify the sender's signature. This is not the case for signcryption, since the confidentiality property implies that only the recipient should be able to read the contents of a signcrypted message sent to him. However, it is possible to achieve non-repudiation by other means.

An, Dodis, and Rabin [1] argue that non-repudiation should not be included in the definition of signcryption, since there are some cases where it is explicitly undesirable. For example, Krawczyk and Rabin [6] have proposed chameleon signature schemes, in which *only* the intended recipient of a message can verify its authenticity. To allow such possibilities also in the case of signcryption, we will not include non-repudiation in our definition. However, when discussing the various proposed signcryption schemes, we indicate which of them provide non-repudiation.

Among those signcryption schemes which provide non-repudiation, some provide it in the following way: The recipient of a signcrypted message recovers the original message m, along with a signature s of that message under some signature scheme $\mathcal{S}$, and provides (m, s) to the verifier, who verifies the sender's signature using $\mathcal{S}$. Such a signcryption scheme is called *$\mathcal{S}$-verifiable*, since the verifier uses only the signature scheme $\mathcal{S}$. This term was introduced by Shin, Lee, and Shim [14], who proposed the first DSA-verifiable signcryption scheme.

$\mathcal{S}$-verifiable signcryption schemes are especially desirable if $\mathcal{S}$ is a commonly used signature scheme, as is the case with DSA. We will indicate which of the proposed signcryption schemes are $\mathcal{S}$-verifiable, and under which signature scheme $\mathcal{S}$.

2.3 Forward Secrecy and Past Recovery

By *forward secrecy*, we mean that an attacker cannot read signcrypted messages, even with access to the sender's private key. That is, the confidentiality of signcrypted messages is protected, even if the sender's private key is compromised.

On the other hand, the sender of a signcrypted message may later want to recover its contents. The sender could simply store a copy of the original message, but this would require additional storage and could be a security risk. Some signcryption schemes offer a better alternative; they allow the sender to use his private key to recover the original message from the cipher-

text. (Note that traditional encryption schemes do not offer this possibility, since the sender's key pair is not involved in the encryption process.) Those signcryption schemes which offer this possibility are said to provide *past recovery*. This property was first considered by Zheng [15], whose original signcryption scheme provides it.

Clearly forward secrecy and past recovery are mutually exclusive. Thus schemes which provide past recovery should be used with care, since the advantage of past recovery comes at the cost of losing forward secrecy. We will leave enough flexibility in our security model to allow for both forward secrecy and past recovery. We will indicate which of them is provided by each of the proposed signcryption schemes.

Chapter 3

Security Models and Generic Constructions

In this chapter, we present the formal definition and security model for signcryption given by An, Dodis, and Rabin [1]. Their model is closely related to the security models for encryption and signature schemes, which we also present. We then present several signcryption schemes which are provably secure under their model. These constructions are generic, in the sense that they can be built from any encryption and signature schemes satisfying certain security requirements.

3.1 Encryption Schemes

3.1.1 Syntax

An *encryption scheme* is a triple of algorithms $\mathcal{E} = (\mathsf{Enc\text{-}KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Enc\text{-}KeyGen}$ is a randomized algorithm which accepts 1^k, where k is a security parameter[1], and outputs a key pair $(\mathsf{EK}, \mathsf{DK})$. EK is the encryption key, which is made public. DK is the decryption key, which is kept secret. This operation is denoted by $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$.

- Enc is a randomized algorithm which accepts an encryption key EK and a message m, and outputs a ciphertext c. This operation is denoted

[1] In this context, 1^k denotes a string of k ones. The algorithms $\mathsf{Enc\text{-}KeyGen}$, Enc, and Dec must run in polynomial time with respect to the length of this parameter, i.e. $|1^k| = k$. We will also use k later in the definition of security.

by $c \leftarrow \mathsf{Enc}_{\mathsf{EK}}(m)$, or simply $c \leftarrow \mathsf{Enc}(m)$ if it is clear which encryption key is being used.

- Dec is a deterministic algorithm which accepts a decryption key DK and a ciphertext c, and outputs a message m, or the special symbol $\perp$ in the case of an invalid ciphertext. This operation is denoted by $m \leftarrow \mathsf{Dec}_{\mathsf{EK}}(c)$, or simply $m \leftarrow \mathsf{Dec}(c)$ if it is clear which decryption key is being used.

We require that $\mathsf{Dec}_{\mathsf{DK}}(\mathsf{Enc}_{\mathsf{EK}}(m)) = m$ for all m, and for all key pairs $(\mathsf{EK}, \mathsf{DK})$ generated by Enc-KeyGen.

3.1.2 Security

To define the security of an encryption scheme, we must specify two things: the goal of the adversary, and the capabilities given to the adversary. The highest level of security is provided when an adversary is unable to achieve even the weakest goal, when given the strongest capabilities.

We will always define the goal of the adversary in terms of a game which the adversary attempts to win. Steps which the adversary must perform will be written in the form

$$(o_1, \ldots, o_m) \leftarrow \mathcal{A}(i_1, \ldots, i_n)$$

where $\mathcal{A}$ is the adversary, $(i_1, \ldots, i_n)$ are the inputs given to it, and $(o_1, \ldots, o_m)$ are the outputs it returns. Unless otherwise stated, the adversary only has access to the information which is passed to it as input.

In the case of encryption, we will take the goal of the adversary to be *distinguishing ciphertexts*. The game played by the adversary consists of two stages. First the adversary is given the encryption key from a fresh key pair, and must select two messages m_0 and m_1 of equal length (the find stage). Then one of the messages is chosen at random and encrypted, and the adversary must correctly determine which one was encrypted (the guess stage). The formal definition of this game is as follows:

1. $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
2. $(m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathsf{EK}, \mathsf{find})$
3. $b \leftarrow^R \{0, 1\}$
4. $c \leftarrow \mathsf{Enc}(m_b)$
5. $\tilde{b} \leftarrow \mathcal{A}(c, \alpha, \mathsf{guess})$.

$\mathcal{A}$ wins in the case that $b = \tilde{b}$. The adversary is permitted to store some state information α between the find and guess stages.

Of course, $\mathcal{A}$ could win this game with probability $1/2$ by simply choosing $\tilde{b}$ at random. Thus we consider an adversary successful only if it can win the game with probability significantly greater than $1/2$. To make this notion more precise, we use the following definition.

Definition 3.1. *We say that a function $f : \mathbb{N} \to [0, 1]$ is* negligible *if, for all $c > 0$, there exists $k_0 \in \mathbb{N}$ such that $f(k) \leq \frac{1}{k^c}$ whenever $k \geq k_0$. We use the notation $\mathsf{negl}(k)$ to indicate "some negligible function of k".*

Now we can complete the definition of indistinguishability. If $\mathcal{E}$ is an encryption scheme for which

$$\Pr[\mathcal{A} \text{ wins}] < \frac{1}{2} + \mathsf{negl}(k)$$

for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, then we say that $\mathcal{E}$ achieves *indistinguishability of ciphertexts*, or IND.

Next we must define the powers given to $\mathcal{A}$. At the very least, $\mathcal{A}$ has access to the encryption key EK, allowing it to encrypt arbitrary messages. This type of attack is known as a *chosen plaintext attack* or CPA. A stronger type of attack is the *chosen ciphertext attack* or CCA2, in which $\mathcal{A}$ may decrypt arbitrary ciphertexts via a decryption oracle[2], with the exception that it may not decrypt the target ciphertext c given to it in step 4 of the game.

When indicating the security level provided by an encryption scheme, we write the adversarial goal and capabilities together. For example, an encryption scheme which achieves indistinguishability of ciphertexts against a chosen ciphertext attack is said to be IND-CCA2-secure.

Generalizing the chosen ciphertext attack

In fact, the chosen ciphertext attack can be slightly too restrictive. To see this, consider taking an IND-CCA2-secure encryption scheme, and modifying it so the encryption algorithm appends a random bit to the ciphertext after encryption, and the decryption algorithm discards this bit before decryption. Intuitively, appending a random bit should not affect the security of the scheme. But in fact the resulting scheme is not IND-CCA2-secure, since the

[2]The digit 2 in CCA2 indicates that the decryption oracle may be queried in both the find and guess stages. This is in contrast to CCA1, in which the decryption oracle is only available during the find stage.

adversary can simply toggle the random bit of the target ciphertext, then query the decryption oracle with this "new" ciphertext.

This problem led An, Dodis, and Rabin [1] to generalize CCA2 slightly, by introducing an equivalence relation $\mathcal{R}(\cdot,\cdot)$ on ciphertexts, and prohibiting the attacker from querying the decryption oracle with any ciphertext equivalent to the target ciphertext under $\mathcal{R}$.

To ensure that this generalization weakens the attacker's capabilities as little as possible, we introduce some restrictions on $\mathcal{R}$. First of all, the relation must be efficient to evaluate, given only the encryption key EK. In addition, it must be *decryption-respecting*, which means that $\mathcal{R}(c_1, c_2) = \mathsf{true}$ only if $\mathsf{Dec}(c_1) = \mathsf{Dec}(c_2)$. That is, we may only restrict the attacker from decrypting other encryptions of the target message.

We say that an encryption scheme $\mathcal{E}$ is secure against a *generalized chosen ciphertext attack* or gCCA2 if there is some equivalence relation $\mathcal{R}$, which satisfies the above conditions, and with respect to which $\mathcal{E}$ is CCA2-secure. Clearly any CCA2-secure encryption scheme is also gCCA2-secure, since we can simply take $\mathcal{R}$ to be the ordinary equality relation. Also note that appending a random bit to an IND-CCA2-secure encryption scheme is no longer a problem, since the resulting scheme is still IND-gCCA2-secure. To see this, simply define $\mathcal{R}(c_1, c_2)$ to be true whenever c_1 and c_2 agree on all but the appended bit.

3.2 Signature Schemes

3.2.1 Syntax

A *signature scheme* is a triple of algorithms $\mathcal{S} = (\mathsf{Sig\text{-}KeyGen}, \mathsf{Sig}, \mathsf{Ver})$:

- Sig-KeyGen is a randomized algorithm which accepts 1^k, where k is a security parameter, and outputs a key pair $(\mathsf{SK}, \mathsf{VK})$. SK is the signing key, which is kept private. VK is the verification key, which is made public. This operation is denoted by $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$.

- Sig is a randomized algorithm which accepts a signing key SK and a message m, and outputs a signature s. This operation is denoted by $s \leftarrow \mathsf{Sig}_{\mathsf{SK}}(m)$, or simply $s \leftarrow \mathsf{Sig}(m)$ if it is clear which signing key is being used.

- Ver is a deterministic algorithm which accepts a verification key VK, a message m and a signature s, and outputs an answer a, which is either succeed or fail. This operation is denoted by $a \leftarrow \mathsf{Ver}_{\mathsf{VK}}(m, s)$,

or simply $a \leftarrow \mathsf{Ver}(m, s)$ if it is clear which verification key is being used.

We require that $\mathsf{Ver}_{\mathsf{VK}}(m, \mathsf{Sig}_{\mathsf{SK}}(m)) = \mathsf{succeed}$ for all m, and for all key pairs $(\mathsf{SK}, \mathsf{VK})$ generated by Sig-KeyGen.

3.2.2 Security

As in the case of encryption schemes, we must specify both the goal of the adversary, and the capabilities given to it.

As the goal of the adversary, we will use *existential forgery*. To achieve existential forgery, the adversary $\mathcal{A}$ must simply produce a valid signature, for any message of its choosing. That is, it must win the following game:

1. $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
2. $(m, s) \leftarrow \mathcal{A}(\mathsf{VK})$.

$\mathcal{A}$ wins the game if $\mathsf{Ver}_{\mathsf{VK}}(m, s)$ returns $\mathsf{succeed}$. If $\mathcal{S}$ is a signature scheme for which

$$\Pr[\mathcal{A} \text{ wins}] < \mathsf{negl}(k)$$

for all PPT adversaries $\mathcal{A}$, then we say that $\mathcal{S}$ achieves *existential unforgeability*, or UF.

Next we must define the powers given to $\mathcal{A}$. In a *no message attack* or NMA, the adversary $\mathcal{A}$ is only given access to the verification key VK. In a *chosen message attack* or CMA, the adversary is additionally given access to a signing oracle. In this case, $\mathcal{A}$ must produce a signature for a "new" message, i.e. one which it did not previously sign with the signing oracle.

As with encryption schemes, to indicate the security level provided by a signature scheme we write the adversarial goal and capabilities together. For example, a signature scheme which achieves existential unforgeability against a chosen message attack is said to be UF-CMA-secure.

Although UF-CMA-security is sufficient for almost all applications, it can be strengthened slightly, by declaring the adversary successful if it produces either a signature for a "new" message (as before), or else a "new" signature for a previously signed message. This type of security is known as *strong unforgeability* against a chosen message attack, or sUF-CMA.

3.3 Signcryption

For simplicity, we first present the definition of signcryption in the so-called *two-user setting*. In this setting, we assume that only two users, a sender A

and a recipient B are using the signcryption scheme. Thus we do not need to worry about attacks such as identity fraud, where one user pretends to be another. Later we will extend the definition to the *multi-user setting*, which protects against such attacks.

3.3.1 Syntax

A *signcryption scheme* is a triple of algorithms $\mathcal{SC} = (\mathsf{KeyGen}, \mathsf{SigEnc}, \mathsf{VerDec})$:

- KeyGen is a randomized algorithm which accepts 1^k, where k is a security parameter, and outputs a key pair $(\mathsf{SDK}, \mathsf{VEK})$. SDK is the signing and decryption key, which is kept secret. VEK is the verification and encryption key, which is made public. This operation is denoted by $(\mathsf{SDK}, \mathsf{VEK}) \leftarrow \mathsf{KeyGen}(1^k)$. Both the sender A and the recipient B of a message must invoke $\mathsf{Sig\text{-}KeyGen}$ before they can communicate; we call their key pairs $(\mathsf{SDK}_A, \mathsf{VEK}_A)$ and $(\mathsf{SDK}_B, \mathsf{VEK}_B)$ respectively.

- SigEnc is a randomized algorithm which accepts the sender's signing and decryption key SDK_A and the recipient's verification and encryption key VEK_B, as well as a message m, and outputs a signcrypted message u. This operation is denoted by $u \leftarrow \mathsf{SigEnc}_{\mathsf{SDK}_A,\mathsf{VEK}_B}(m)$, or simply $s \leftarrow \mathsf{SigEnc}(m)$ if it is clear which keys are being used.

- VerDec is a deterministic algorithm which accepts the sender's verification and encryption key VEK_A and the recipient's signing and decryption key SDK_B, as well as a signcrypted message u, and outputs a message m, or the special symbol $\perp$ in the case of an invalid signcryption. This operation is denoted by $m \leftarrow \mathsf{VerDec}_{\mathsf{VEK}_A,\mathsf{SDK}_B}(u)$, or simply $m \leftarrow \mathsf{VerDec}(u)$ if it is clear which keys are being used.

We require that $\mathsf{VerDec}_{\mathsf{VEK}_A,\mathsf{SDK}_B}(\mathsf{SigEnc}_{\mathsf{SDK}_A,\mathsf{VEK}_B}(m)) = m$ for all m, and for all key pairs $(\mathsf{SDK}_A, \mathsf{VEK}_A)$ and $(\mathsf{SDK}_B, \mathsf{VEK}_B)$ generated by KeyGen.

3.3.2 Security

In defining security for signcryption, we distinguish between two classes of attacks: *outsider attacks*, in which the attacker has access only to public information, and *insider attacks*, in which the attacker additionally has either the sender's or the recipient's private key. In each case, we must ensure that the two basic security goals of signcryption, confidentiality and authenticity, are provided.

Outsider Security

In the outsider security model, we assume that the adversary $\mathcal{A}$ only has access to public information, i.e. $pub = \{\mathsf{VEK}_A, \mathsf{VEK}_B\}$. The goal of the adversary is to break either the confidentiality of the scheme, by distinguishing ciphertexts, or the authenticity of the scheme, by achieving existential forgery. In both cases, we give $\mathcal{A}$ oracle access to the sender's signcryption functionality, $\mathsf{SigEnc}_{\mathsf{SDK}_A,\mathsf{VEK}_B}(\cdot)$, and the recipient's de-signcryption functionality, $\mathsf{VerDec}_{\mathsf{VEK}_A,\mathsf{SDK}_B}(\cdot)$.

To distinguish ciphertexts, the adversary $\mathcal{A}$ must win the following game, with probability significantly greater than 1/2:

1. $(\mathsf{SDK}_A, \mathsf{VEK}_A) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(\mathsf{SDK}_B, \mathsf{VEK}_B) \leftarrow \mathsf{KeyGen}(1^k)$
3. $(m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathsf{VEK}_A, \mathsf{VEK}_B, \mathsf{find})$
4. $b \leftarrow^R \{0, 1\}$
5. $u \leftarrow \mathsf{SigEnc}(m_b)$
6. $\tilde{b} \leftarrow \mathcal{A}(u, \alpha, \mathsf{guess})$

$\mathcal{A}$ wins in the case that $b = \tilde{b}$. It is allowed to make arbitrary queries to its signcryption and de-signcryption oracles at any point during its find and guess stages (steps 3 and 6), with the exception that in the guess stage we disallow it from querying the de-signcryption oracle with the target ciphertext u given to it in step 5. As with encryption schemes, we require that $|m_0| = |m_1|$.

Note that this is completely analogous to distinguishing ciphertexts in the case of encryption schemes. If $\mathcal{SC}$ is a signcryption scheme for which

$$\Pr[\mathcal{A} \text{ wins}] < \frac{1}{2} + \mathsf{negl}(k)$$

for all PPT adversaries $\mathcal{A}$, then we say $\mathcal{SC}$ is *outsider-secure* against an IND-CCA2 attack.

As in the case of encryption schemes, security under IND-CCA2 attacks can be too restrictive. Thus we introduce a decryption-respecting[3] equivalence relation $\mathcal{R}$, which can be efficiently evaluated using only the public information, VEK_A and VEK_B. We disallow $\mathcal{A}$ from querying the de-signcryption oracle with any ciphertext equivalent to the target ciphertext under $\mathcal{R}$. We say that $\mathcal{SC}$ is outsider-secure against an IND-gCCA2 attack if

[3]In this case, decryption-respecting means that $\mathcal{R}(u_1, u_2) = \mathsf{true}$ only if $\mathsf{VerDec}(u_1) = \mathsf{VerDec}(u_2)$.

there is some equivalence relation $\mathcal{R}$ with respect to which $\mathcal{SC}$ is outsider-secure against an IND-CCA2 attack.

To achieve existential forgery, the adversary $\mathcal{A}$ must win the following game, with probability significantly greater than zero:

1. $(\mathsf{SDK}_A, \mathsf{VEK}_A) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(\mathsf{SDK}_B, \mathsf{VEK}_B) \leftarrow \mathsf{KeyGen}(1^k)$
3. $u \leftarrow \mathcal{A}(\mathsf{VEK}_A, \mathsf{VEK}_B)$

$\mathcal{A}$ wins the game if u is a valid signcryption of any message not queried to the signcryption oracle.

Note that this is very similar to achieving existential forgery in the case of signature schemes, except that the adversary here is not required to know which message m it has forged. If $\mathcal{SC}$ is a signcryption scheme for which

$$\Pr[\mathcal{A} \text{ wins}] < \mathsf{negl}(k)$$

for all PPT adversaries $\mathcal{A}$, then we say $\mathcal{SC}$ is *outsider-secure* against a UF-CMA attack. As in the case of signature schemes, we can define strong unforgeability against a chosen message attack, or sUF-CMA, by modifying the game so that $\mathcal{A}$ wins as long as u is different from any of the ciphertexts returned by the signcryption oracle.

In the context of outsider security, the suffixes CCA2 (in IND-CCA2 and IND-gCCA2) and CMA (in UF-CMA and sUF-CMA) are slightly misleading, since in all cases the adversary is allowed both to signcrypt chosen messages, and to de-signcrypt chosen ciphertexts, via its oracles. Fortunately this problem disappears when we move to insider security.

Insider Security

To define insider security, we simply make some small modifications to the definitions given above.

When distinguishing ciphertexts, we give the adversary $\mathcal{A}$ access to the sender's secret key, SDK_A. That is, step 3 of the game becomes

$$(m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathsf{SDK}_A, \mathsf{VEK}_A, \mathsf{VEK}_B, \mathsf{find}).$$

Of course, with access to SDK_A, the adversary no longer needs the signcryption oracle $\mathsf{SigEnc}_{\mathsf{SDK}_A,\mathsf{VEK}_B}(\cdot)$. Even though we give the adversary access to SDK_A, we still allow the relation $\mathcal{R}$ to depend only on VEK_A and VEK_B in the case of an IND-gCCA2 attacks.

When producing an existential forgery, we give $\mathcal{A}$ access to the recipient's secret key, SDK_B. That is, step 3 of the game becomes

$$u \leftarrow \mathcal{A}(\mathsf{VEK}_A, \mathsf{SDK}_B, \mathsf{VEK}_B).$$

With access to SDK_B, the adversary no longer needs the de-signcryption oracle $\mathsf{VerDec}_{\mathsf{VEK}_A,\mathsf{SDK}_B}(\cdot)$.

Optimally, we would like a signcryption scheme to be secure against IND-CCA2 attacks on its confidentiality and against sUF-CMA attacks on its authenticity. However, security against IND-gCCA2 and UF-CMA attacks is sufficient for most applications.

Insider vs. Outsider Security

Since insider security is strictly stronger than outsider security, and since there are already many signcryption schemes which provide insider security, one might wonder why it is useful to define outsider security at all. One possible reason is that insider security on the confidentiality of a signcryption scheme is precisely forward secrecy, and this precludes past recovery (see Section 2.3).

In the case of authenticity, insider security is necessary for any scheme which provides non-repudiation; without it, there would be nothing to prevent a user from forging a message to himself. It is not clear whether there are any situations where insider security on authenticity would be explicitly undesirable, and indeed all the signcryption schemes we present here provide insider security on authenticity.

3.3.3 The Multi-User Setting

Now that we have defined signcryption in the two-user setting, we can make some small modifications to define it in the multi-user setting. In this setting, there may be any number of users who wish to communicate with each other. We assume that each user P has an identity ID_P, which members of the network can use to uniquely identify the user (for example by name or email address) and obtain P's public key VEK_P. In practice, ID_P and VEK_P are typically contained in a certificate provided by a certification authority.

Note that multi-user settings have been defined for other cryptographic primitives. For example, Menezes and Smart [8] have defined a multi-user setting for signature schemes.

Syntax

To begin, we modify the signcryption and de-signcryption algorithms, SigEnc and VerDec. In the two-user setting we defined these to accept the keys of the sender and recipient. In the multi-user setting they accept identities instead of keys.

For a given user A who wishes to send a message to some user B', SigEnc takes as input a message m, and two identities $\mathsf{ID}_{A'}$ and $\mathsf{ID}_{B'}$. After checking that $\mathsf{ID}_{A'} = \mathsf{ID}_A$, SigEnc then signcrypts the message m using SDK_A and $\mathsf{VEK}_{B'}$, and outputs $(u, \mathsf{ID}_A, \mathsf{ID}_{B'})$. This operation is denoted by $(u, \mathsf{ID}_A, \mathsf{ID}_{B'}) \leftarrow \mathsf{SigEnc}_A(m, \mathsf{ID}_{A'}, \mathsf{ID}_{B'})$. When describing signcryption schemes, we typically omit the identity verification step, and simply write $u \leftarrow \mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_{B'})$.

For a given user B who has received a ciphertext from some user A', VerDec takes as input a ciphertext u, and two identities $\mathsf{ID}_{A'}$ and $\mathsf{ID}_{B'}$. After checking that $\mathsf{ID}_{B'} = \mathsf{ID}_B$, VerDec then de-signcrypts the ciphertext u using $\mathsf{VEK}_{A'}$ and SDK_B. This operation is denoted by $m \leftarrow \mathsf{VerDec}_B(u, \mathsf{ID}_{A'}, \mathsf{ID}_{B'})$. Once again, the identity verification step is typically omitted, and we simply write $m \leftarrow \mathsf{VerDec}(m, \mathsf{ID}_{A'}, \mathsf{ID}_B)$.

Security

As in the two-user setting, we distinguish between outsider and insider security. In both cases, we define security as in the two-user setting, but with a few small modifications.

In the outsider security model, when attacking communication between a pair of users A and B, we assume that the adversary $\mathcal{A}$ has access to the private keys of all users other than A and B. As in the two-user setting, we give $\mathcal{A}$ oracle access to A's signcryption functionality and B's de-signcryption functionality. However, we make the oracles somewhat more flexible in the multi-user setting, by allowing $\mathcal{A}$ to select any recipient (not necessarily B) when calling the signcryption oracle, and any sender (not necessarily A) when calling the de-signcryption oracle. That is, $\mathcal{A}$ has oracle access to $\mathsf{SigEnc}(\cdot, \mathsf{ID}_A, \cdot)$ and $\mathsf{VerDec}(\cdot, \cdot, \mathsf{ID}_B)$.

To break the $\mathsf{UF\text{-}CMA}$-security, $\mathcal{A}$ must produce a valid signcryption $(u, \mathsf{ID}_A, \mathsf{ID}_B)$ of a message m, without asking the signcryption oracle to signcrypt $(m, \mathsf{ID}_A, \mathsf{ID}_B)$. Note, however, that $\mathcal{A}$ may request the signcryption of $(m, \mathsf{ID}_A, \mathsf{ID}_{B'})$ for some $B' \neq B$. Now the game is as follows:

1. $(\mathsf{SDK}_A, \mathsf{VEK}_A) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(\mathsf{SDK}_B, \mathsf{VEK}_B) \leftarrow \mathsf{KeyGen}(1^k)$

3. $u \leftarrow \mathcal{A}(\mathsf{VEK}_A, \mathsf{VEK}_B)$

$\mathcal{A}$ wins the game if $\mathsf{VerDec}(u, \mathsf{ID}_A, \mathsf{ID}_B)$ returns anything other than $\perp$. The fact that the attacker has access to the private keys of all other users is modelled by allowing $\mathcal{A}$ to create its own keys pairs and corresponding identities, and pass them to the oracles as parameters. $\mathcal{A}$ is not required to generate key pairs via KeyGen, thus giving it a great deal of freedom in terms of oracle access.

To break the IND-CCA2-security, $\mathcal{A}$ must produce two messages m_0 and m_1, and then distinguish between $\mathsf{SigEnc}(m_0, \mathsf{ID}_A, \mathsf{ID}_B)$ and $\mathsf{SigEnc}(m_1, \mathsf{ID}_A, \mathsf{ID}_B)$ without passing $(u, \mathsf{ID}_A, \mathsf{ID}_B)$ to the de-signcryption oracle, where u is the challenge ciphertext. Note, however, that $\mathcal{A}$ may request the de-signcryption of $(u, \mathsf{ID}_{A'}, \mathsf{ID}_B)$ for some $A' \neq A$. More formally, $\mathcal{A}$ must win the following game:

1. $(\mathsf{SDK}_A, \mathsf{VEK}_A) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(\mathsf{SDK}_B, \mathsf{VEK}_B) \leftarrow \mathsf{KeyGen}(1^k)$
3. $(m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathsf{VEK}_A, \mathsf{VEK}_B, \mathsf{find})$
4. $b \leftarrow^R \{0, 1\}$
5. $u \leftarrow \mathsf{SigEnc}_A(m_b, \mathsf{ID}_A, \mathsf{ID}_B)$
6. $\tilde{b} \leftarrow \mathcal{A}(u, \alpha, \mathsf{guess})$

The adversary $\mathcal{A}$ wins in the case that $b = \tilde{b}$. As usual, we require that $|m_0| = |m_1|$.

In the insider security model, the adversary $\mathcal{A}$ attacks either the authenticity of a sender A, or the confidentiality of a recipient B. In the first case, we give $\mathcal{A}$ access to the private keys of all users except A. Also $\mathcal{A}$ has access to A's signcryption oracle, $\mathsf{SigEnc}(\cdot, \mathsf{ID}_A, \cdot)$. It must create a valid signcryption $(u, \mathsf{ID}_A, \mathsf{ID}_{B'})$ for some message m and recipient B', without querying its signcryption oracle with $(m, \mathsf{ID}_A, \mathsf{ID}_{B'})$. As before, it may make the query $(m, \mathsf{ID}_A, \mathsf{ID}_{B''})$ for some $B'' \neq B'$. The game played by $\mathcal{A}$ is:

1. $(\mathsf{SDK}_A, \mathsf{VEK}_A) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(u, \mathsf{ID}_{B'}) \leftarrow \mathcal{A}(\mathsf{VEK}_A)$

$\mathcal{A}$ wins the game if $\mathsf{VerDec}(u, \mathsf{ID}_A, \mathsf{ID}_{B'})$ returns anything other than $\perp$. Note that only A's key pair is generated at the start, leaving $\mathcal{A}$ free to choose keys for B' as it pleases. Once again, it is not required to do so using KeyGen.

When attacking the confidentiality of a user B, we give $\mathcal{A}$ access to the private keys of all users except B. Also, $\mathcal{A}$ has access to B's de-signcryption

oracle, $\mathsf{VerDec}(\cdot, \cdot, \mathsf{ID}_B)$. It must choose a sender A', produce two messages m_0 and m_1, and then distinguish between $\mathsf{SigEnc}(m_0, \mathsf{ID}_{A'}, \mathsf{ID}_B)$ and $\mathsf{SigEnc}(m_1, \mathsf{ID}_{A'}, \mathsf{ID}_B)$ without passing $(u, \mathsf{ID}_{A'}, \mathsf{ID}_B)$ to its de-signcryption oracle, where u is the challenge ciphertext. Of course, it may make the query $(u, \mathsf{ID}_{A''}, \mathsf{ID}_B)$ for some $A'' \neq A'$. The game played by $\mathcal{A}$ is:

1. $(\mathsf{SDK}_B, \mathsf{VEK}_B) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(m_0, m_1, \mathsf{ID}_{A'}, \alpha) \leftarrow \mathcal{A}(\mathsf{VEK}_B, \mathsf{find})$
3. $b \leftarrow^R \{0, 1\}$
4. $u \leftarrow \mathsf{SigEnc}(m_b, \mathsf{ID}_{A'}, \mathsf{ID}_B)$
5. $\tilde{b} \leftarrow \mathcal{A}(u, \alpha, \mathsf{guess})$

As usual, $\mathcal{A}$ wins in the case that $b = \tilde{b}$. Note that only B's key pair is generated at the start, leaving $\mathcal{A}$ free to choose keys for A' as it pleases.

Once again, we can define the IND-gCCA2 security level for confidentiality, and the sUF-CMA security level for authenticity. In the case of IND-gCCA2 security, the relation $\mathcal{R}$ must take into account the identities of the sender and recipient to be decryption-respecting. Specifically, we must have $\mathcal{R}(u_1, u_2) = \mathsf{true}$ only if u_1 and u_2 are both signcryptions of the same message, with the same sender and recipient.

3.4 Encrypt-then-Sign and Sign-then-Encrypt

Prior to Zheng's work [15], the usual way to provide both confidentiality and authenticity was to sequentially compose encryption and signature schemes. This can be done in two ways: encrypt-then-sign ($\mathcal{E}t\mathcal{S}$), and sign-then-encrypt ($\mathcal{S}t\mathcal{E}$).

To verify that our definition of signcryption is a good one, it is worthwhile to start by checking whether these two constructions are secure as signcryption schemes. We begin by defining $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ in the two-user setting, and then adapt the definitions to provide security in the multi-user setting.

Definition 3.2 (Encrypt-then-sign). *Given an encryption scheme $\mathcal{E}$ and a signature scheme $\mathcal{S}$, we define the two-user signcryption scheme $\mathcal{E}t\mathcal{S}$ as follows:*

- $\mathsf{KeyGen}(1^k)$*:*
  1. $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
  2. $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$

3. *Return* $(\mathsf{SDK}, \mathsf{VEK}) \leftarrow ((\mathsf{SK}, \mathsf{DK}), (\mathsf{VK}, \mathsf{EK}))$.

- $\mathsf{SigEnc}_{\mathsf{SDK}_A, \mathsf{VEK}_B}(m)$*:*
  1. $e \leftarrow \mathsf{Enc}_{\mathsf{EK}_B}(m)$
  2. $s \leftarrow \mathsf{Sig}_{\mathsf{SK}_A}(e)$
  3. *Return* (e, s).
- $\mathsf{VerDec}_{\mathsf{VEK}_A, \mathsf{SDK}_B}(e, s)$*:*
  1. $a \leftarrow \mathsf{Ver}_{\mathsf{VK}_A}(e, s)$
  2. *If* $a = \mathsf{succeed}$ *return* $m \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(e)$ *else return* $\perp$.

Definition 3.3 (Sign-then-encrypt). *Given an encryption scheme* $\mathcal{E}$ *and a signature scheme* $\mathcal{S}$*, we define the two-user signcryption scheme* $\mathcal{S}t\mathcal{E}$ *as follows:*

- $\mathsf{KeyGen}(1^k)$*:*
  1. $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
  2. $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
  3. *Return* $(\mathsf{SDK}, \mathsf{VEK}) \leftarrow ((\mathsf{SK}, \mathsf{DK}), (\mathsf{VK}, \mathsf{EK}))$.
- $\mathsf{SigEnc}_{\mathsf{SDK}_A, \mathsf{VEK}_B}(m)$*:*
  1. $s \leftarrow \mathsf{Sig}_{\mathsf{SK}_A}(m)$
  2. *Return* $u \leftarrow \mathsf{Enc}_{\mathsf{EK}_B}(m||s)$.
- $\mathsf{VerDec}_{\mathsf{VEK}_A, \mathsf{SDK}_B}(u)$*:*
  1. $m||s \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(u)$
  2. *If* $\mathsf{Ver}_{\mathsf{VK}_A}(m, s) = \mathsf{succeed}$ *return* m *else return* $\perp$.

Theorem 3.4. *Suppose that* $\mathcal{E}$ *is an* IND-gCCA2*-secure encryption scheme, and that* $\mathcal{S}$ *is a* UF-CMA*-secure signature scheme. Then* $\mathcal{E}t\mathcal{S}$ *and* $\mathcal{S}t\mathcal{E}$ *are insider-secure against* IND-gCCA2 *attacks on confidentiality, and against* UF-CMA *attacks on authenticity, in the two-user setting.*

The proof is given in An, Dodis, and Rabin [1]. We omit it here, and instead we give the proofs for the corresponding theorems in the multi-user setting.

Now we consider how to adapt $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ to the multi-user setting. Note that the current definitions cannot be left unchanged, since they are insecure even against outsider attacks in the multi-user setting. In the case of $\mathcal{E}t\mathcal{S}$ an adversary can distinguish between ciphertexts from A to B by removing A's signature s from the challenge ciphertext $u = (e, s)$, and replacing it with the signature s' of some other user A'. It can then de-signcrypt (e, s')

using its de-signcryption oracle, revealing the message. In the case of $\mathcal{S}t\mathcal{E}$ an adversary can forge a message from A to B by choosing some message m, requesting $u' \leftarrow \mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_{B'})$ for some other user B', then decrypting u' into $m||s$ and re-encrypting it with B's public key, to get a valid signcryption u from A to B.

To avoid these attacks, we need to bring the identities of the sender and recipient into the calculations. An, Dodis, and Rabin [1] solved this problem by simply appending the sender's identity before encrypting, and appending the recipient's identity before signing. This results in the following schemes:

Definition 3.5 (Multi-user encrypt-then-sign). *Given an encryption scheme $\mathcal{E}$ and a signature scheme $\mathcal{S}$, we define the multi-user signcryption scheme $\mathcal{E}t\mathcal{S}'$ as follows:*

- $\mathsf{KeyGen}(1^k)$*:*
  1. $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
  2. $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
  3. *Return* $(\mathsf{SDK}, \mathsf{VEK}) \leftarrow ((\mathsf{SK}, \mathsf{DK}), (\mathsf{VK}, \mathsf{EK}))$.
- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$*:*
  1. $e \leftarrow \mathsf{Enc}_{\mathsf{EK}_B}(m||\mathsf{ID}_A)$
  2. $s \leftarrow \mathsf{Sig}_{\mathsf{SK}_A}(e||\mathsf{ID}_B)$
  3. *Return* (e, s).
- $\mathsf{VerDec}(e, s, \mathsf{ID}_A, \mathsf{ID}_B)$*:*
  1. *If* $\mathsf{Ver}_{\mathsf{VK}_A}(e||\mathsf{ID}_B, s) = \mathsf{fail}$ *return* $\perp$.
  2. $m||\mathsf{ID}_{A'} \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(e)$
  3. *If* $\mathsf{ID}_{A'} = \mathsf{ID}_A$ *return* m *else return* $\perp$.

Definition 3.6 (Multi-user sign-then-encrypt). *Given an encryption scheme $\mathcal{E}$ and a signature scheme $\mathcal{S}$, we define the multi-user signcryption scheme $\mathcal{S}t\mathcal{E}'$ as follows:*

- $\mathsf{KeyGen}(1^k)$*:*
  1. $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
  2. $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
  3. *Return* $(\mathsf{SDK}, \mathsf{VEK}) \leftarrow ((\mathsf{SK}, \mathsf{DK}), (\mathsf{VK}, \mathsf{EK}))$.
- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$*:*
  1. $s \leftarrow \mathsf{Sig}_{\mathsf{SK}_A}(m||\mathsf{ID}_B)$
  2. *Return* $u \leftarrow \mathsf{Enc}_{\mathsf{EK}_B}(m||s||\mathsf{ID}_A)$.

- $\mathsf{VerDec}(u, \mathsf{ID}_A, \mathsf{ID}_B)$*:*
  1. $m||s||\mathsf{ID}_{A'} \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(u)$
  2. *If* $\mathsf{ID}_{A'} \neq \mathsf{ID}_A$ *return* $\perp$.
  3. *If* $\mathsf{Ver}_{\mathsf{VK}_A}(m||\mathsf{ID}_B, s) = \mathsf{succeed}$ *return* m *else return* $\perp$.

These new multi-user versions are indeed secure in the insider security model, as demonstrated by the following theorems. The proofs presented here are the original work of the author.

Theorem 3.7. *Suppose that* $\mathcal{E}$ *is an* IND-gCCA2*-secure encryption scheme, and that* $\mathcal{S}$ *is a* UF-CMA*-secure signature scheme. Then* $\mathcal{E}t\mathcal{S}'$ *is insider-secure against* IND-gCCA2 *attacks on confidentiality, and against* UF-CMA *attacks on authenticity, in the multi-user setting.*

Proof. **Confidentiality:** Since we are dealing with IND-gCCA2 security, we must first deal with the issue of equivalence relations. Let $\mathcal{R}_{\mathcal{E}}$ be the relation with respect to which $\mathcal{E}$ is IND-CCA2-secure, and for $\mathcal{E}t\mathcal{S}'$ define $\mathcal{R}_{\mathcal{E}t\mathcal{S}'}((e_1, s_1, \mathsf{ID}_{A_1}, \mathsf{ID}_{B_1}), (e_2, s_2, \mathsf{ID}_{A_2}, \mathsf{ID}_{B_2}))$ to be $\mathcal{R}_{\mathcal{E}}(e_1, e_2)$ if $A_1 = A_2$, $B_1 = B_2$, $\mathsf{Ver}_{\mathsf{VK}_{A_1}}(e_1||\mathsf{ID}_{B_1}, s_1) = \mathsf{succeed}$, and $\mathsf{Ver}_{\mathsf{VK}_{A_2}}(e_2||\mathsf{ID}_{B_2}, s_2) = \mathsf{succeed}$; or false otherwise. Note that $\mathcal{R}_{\mathcal{E}t\mathcal{S}'}$ can be efficiently evaluated using only VK_{A_1} and VK_{A_2}, which are public.

Now we will show that $\mathcal{E}t\mathcal{S}'$ is insider-secure against IND-CCA2 attacks with respect to $\mathcal{R}_{\mathcal{E}t\mathcal{S}'}$, by way of contradiction. Suppose there exists an adversary $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ which can break the IND-CCA2 security of $\mathcal{E}t\mathcal{S}'$ with non-negligible probability. We convert this to an adversary $\mathcal{A}_{\mathcal{E}}$ which can break the IND-CCA2 security of $\mathcal{E}$ with the same probability, in the same time, and using the same number of oracle queries.

Let $(\mathsf{EK}_B, \mathsf{DK}_B)$ be the key pair which $\mathcal{A}_{\mathcal{E}}$ is trying to attack. In the find stage, after being given EK_B, $\mathcal{A}_{\mathcal{E}}$ does the following:

1. $(\mathsf{SK}_B, \mathsf{VK}_B) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
2. $(m_0, m_1, \mathsf{ID}_{A'}, \alpha) \leftarrow \mathcal{A}_{\mathcal{E}t\mathcal{S}'}((\mathsf{VK}_B, \mathsf{EK}_B), \mathsf{find})$
3. Return $(m_0||\mathsf{ID}_{A'}, m_1||\mathsf{ID}_{A'}, \alpha)$.

In the guess stage, once given the target ciphertext $e = \mathsf{Enc}_{\mathsf{EK}_B}(m_b||\mathsf{ID}_{A'})$ for $b \in_R \{0, 1\}$, $\mathcal{A}_{\mathcal{E}}$ does the following:

1. $s \leftarrow \mathsf{Sig}_{\mathsf{SK}_{A'}}(e||\mathsf{ID}_B)$
2. Return $\tilde{b} \leftarrow \mathcal{A}_{\mathcal{E}t\mathcal{S}'}((e, s, \mathsf{ID}_{A'}, \mathsf{ID}_B), \alpha, \mathsf{guess})$.

Note that $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ may make queries to its de-signcryption oracle. To respond to a query $(e', s', \mathsf{ID}_{A''}, \mathsf{ID}_B)$, $\mathcal{A}_{\mathcal{E}}$ does the following:

1. If $\mathsf{Ver}_{\mathsf{VK}_{A''}}(e'||\mathsf{ID}_B, s') = \mathsf{fail}$ return $\perp$.
2. If $\mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$ return $\perp$.
3. $m||\mathsf{ID}_{A'''} \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(e')$
4. If $\mathsf{ID}_{A'''} = \mathsf{ID}_{A''}$ return m else return $\perp$.

Note that $\mathcal{A}_{\mathcal{E}}$ may perform step 3 using its decryption oracle, since step 2 guarantees that e' is not equivalent to the target ciphertext, e.

To finish the proof, we must check that the above steps perfectly simulate the VerDec algorithm of $\mathcal{E}t\mathcal{S}'$. Apart from step 2, the above steps are precisely those of VerDec. Step 2 could interfere only in the case that $\mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$ (and $\mathsf{Ver}_{\mathsf{VK}_{A''}}(e'||\mathsf{ID}_B, s') = \mathsf{succeed}$, since otherwise step 2 is not executed). Suppose then that $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ makes some oracle query $(e', s', \mathsf{ID}_{A''}, \mathsf{ID}_B)$ for which $\mathsf{Ver}_{\mathsf{VK}_{A''}}(e'||\mathsf{ID}_B, s') = \mathsf{succeed}$ and $\mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$. We know that e is an encryption of $m_b||\mathsf{ID}_{A'}$. Since $\mathcal{R}_{\mathcal{E}}$ is decryption respecting, e' too is an encryption of $m_b||\mathsf{ID}_{A'}$. Now there are two cases to consider. If $\mathsf{ID}_{A''} \neq \mathsf{ID}_{A'}$ then the last step of VerDec would return $\perp$, just as step 2 above. If $\mathsf{ID}_{A''} = \mathsf{ID}_{A'}$ we quickly run into a contradiction. By the definition of s, we have $\mathsf{Ver}_{\mathsf{VK}_{A'}}(e||\mathsf{ID}_B, s) = \mathsf{succeed}$. Since $\mathsf{Ver}_{\mathsf{VK}_{A''}}(e'||\mathsf{ID}_B, s') = \mathsf{succeed}$ as well, we have $\mathcal{R}_{\mathcal{E}t\mathcal{S}'}((e, s, \mathsf{ID}_{A'}, \mathsf{ID}_B), (e', s', \mathsf{ID}_{A''}, \mathsf{ID}_B)) = \mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$. But this contradicts the fact that $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ does not query its de-signcryption oracle with anything equivalent to the target ciphertext.

Authenticity: Suppose there is an adversary $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ which can break the UF-CMA security of $\mathcal{E}t\mathcal{S}'$ with non-negligible probability. We convert this to an adversary $\mathcal{A}_{\mathcal{S}}$ which can break the UF-CMA security of $\mathcal{S}$ with the same probability, in the same time, and using the same number of oracle queries.

Let $(\mathsf{SK}_A, \mathsf{VK}_A)$ be the key pair which $\mathcal{A}_{\mathcal{S}}$ is trying to attack. After being given VK_A, $\mathcal{A}_{\mathcal{S}}$ does the following:

1. $(\mathsf{EK}_A, \mathsf{DK}_A) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
2. $(e, s, \mathsf{ID}_A, \mathsf{ID}_{B'}) \leftarrow \mathcal{A}_{\mathcal{E}t\mathcal{S}'}((\mathsf{VK}_A, \mathsf{EK}_A))$
3. Return $(e||\mathsf{ID}_{B'}, s)$.

In step 2, $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ may make queries to its signcryption oracle. To respond to a query $(m', \mathsf{ID}_A, \mathsf{ID}_{B''})$, $\mathcal{A}_{\mathcal{S}}$ does the following:

1. $e' \leftarrow \mathsf{Enc}_{\mathsf{EK}_{B''}}(m'||\mathsf{ID}_A)$
2. $s' \leftarrow \mathsf{Sig}_{\mathsf{SK}_A}(e'||\mathsf{ID}_{B''})$
3. Return (e', s').

Note that $\mathcal{A}_{\mathcal{S}}$ may perform step 2 using its signing oracle. We must check, however, that the message $e||\mathsf{ID}_{B'}$ signed by $\mathcal{A}_{\mathcal{S}}$ is not equal to any of

the messages $e'||\mathsf{ID}_{B''}$ passed to its signing oracle. Suppose otherwise, i.e. $\mathsf{ID}_{B'} = \mathsf{ID}_{B''}$ and $e = e'$. Note that $e = \mathsf{Enc}_{\mathsf{EK}_{B'}}(m||\mathsf{ID}_A)$ for some m, and $e' = \mathsf{Enc}_{\mathsf{EK}_{B''}}(m'||\mathsf{ID}_A) = \mathsf{Enc}_{\mathsf{EK}_{B'}}(m'||\mathsf{ID}_A)$ for some m'. Then since $e = e'$ we have $\mathsf{Enc}_{\mathsf{EK}_{B'}}(m||\mathsf{ID}_A) = \mathsf{Enc}_{\mathsf{EK}_{B'}}(m'||\mathsf{ID}_A)$, which implies that $m||\mathsf{ID}_A = m'||\mathsf{ID}_A$, so $m = m'$, contradicting the fact that $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ produces a signcryption for a message not queried to its signcryption oracle. □

Theorem 3.8. *Suppose that $\mathcal{E}$ is an* IND-gCCA2*-secure encryption scheme, and that $\mathcal{S}$ is a* UF-CMA*-secure signature scheme. Then $\mathcal{S}t\mathcal{E}'$ is insider-secure against* IND-gCCA2 *attacks on confidentiality, and against* UF-CMA *attacks on authenticity, in the multi-user setting.*

Proof. **Confidentiality:** Since we are dealing with IND-gCCA2 security, we must first deal with the issue of equivalence relations. Let $\mathcal{R}_{\mathcal{E}}$ be the relation with respect to which $\mathcal{E}$ is IND-CCA2-secure, and for $\mathcal{S}t\mathcal{E}'$ define $\mathcal{R}_{\mathcal{S}t\mathcal{E}'}((u_1, \mathsf{ID}_{A_1}, \mathsf{ID}_{B_1}), (u_2, \mathsf{ID}_{A_2}, \mathsf{ID}_{B_2}))$ to be $\mathcal{R}_{\mathcal{E}}(u_1, u_2)$ if if $A_1 = A_2$ and $B_1 = B_2$, or false otherwise.

Now we will show that $\mathcal{S}t\mathcal{E}'$ is insider-secure against IND-CCA2 attacks with respect to $\mathcal{R}_{\mathcal{S}t\mathcal{E}'}$, by way of contradiction. Suppose there exists an adversary $\mathcal{A}_{\mathcal{S}t\mathcal{E}'}$ which can break the IND-CCA2 security of $\mathcal{S}t\mathcal{E}'$ with non-negligible probability. We convert this to an adversary $\mathcal{A}_{\mathcal{E}}$ which can break the IND-CCA2 security of $\mathcal{E}$ with the same probability, in the same time, and using the same number of oracle queries.

Let $(\mathsf{EK}_B, \mathsf{DK}_B)$ be the key pair which $\mathcal{A}_{\mathcal{E}}$ is trying to attack. In the find stage, after being given EK_B, $\mathcal{A}_{\mathcal{E}}$ does the following:

1. $(\mathsf{SK}_B, \mathsf{VK}_B) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
2. $(m_0, m_1, \mathsf{ID}_{A'}, \alpha) \leftarrow \mathcal{A}_{\mathcal{E}t\mathcal{S}'}((\mathsf{VK}_B, \mathsf{EK}_B), \mathsf{find})$
3. $s_0 \leftarrow \mathsf{Sig}_{\mathsf{SK}_{A'}}(m_0||\mathsf{ID}_B)$
4. $s_1 \leftarrow \mathsf{Sig}_{\mathsf{SK}_{A'}}(m_1||\mathsf{ID}_B)$
5. Return $(m_0||s_0||\mathsf{ID}_{A'}, m_1||s_1||\mathsf{ID}_{A'}, \alpha)$.

In the guess stage, once given the target ciphertext $e = \mathsf{Enc}_{\mathsf{EK}_B}(m_b||s_b||\mathsf{ID}_{A'})$ for $b \in_R \{0, 1\}$, $\mathcal{A}_{\mathcal{E}}$ does the following:

1. Return $\tilde{b} \leftarrow \mathcal{A}_{\mathcal{S}t\mathcal{E}'}((e, \mathsf{ID}_{A'}, \mathsf{ID}_B), \alpha, \mathsf{guess})$.

Note that $\mathcal{A}_{\mathcal{S}t\mathcal{E}'}$ may make queries to its de-signcryption oracle. To respond to a query $(e', \mathsf{ID}_{A''}, \mathsf{ID}_B)$, $\mathcal{A}_{\mathcal{E}}$ does the following:

1. If $\mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$ return $\perp$.
2. $m||s||\mathsf{ID}_{A'''} \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(e')$

3. If $\mathsf{ID}_{A'''} \neq \mathsf{ID}_{A''}$ return $\perp$.
4. If $\mathsf{Ver}_{\mathsf{VK}_{A''}}(m||\mathsf{ID}_B, s) = \mathsf{succeed}$ return m else return $\perp$.

Note that $\mathcal{A}_{\mathcal{E}}$ may perform step 2 using its decryption oracle, since step 1 guarantees that e' is not equivalent to the target ciphertext, e.

To finish the proof, we must check that the above steps perfectly simulate the VerDec algorithm of $\mathcal{S}t\mathcal{E}'$. Apart from step 1, the above steps are precisely those of VerDec. Step 1 could interfere only in the case that $\mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$. Suppose then that $\mathcal{A}_{\mathcal{S}t\mathcal{E}'}$ makes some oracle query $(e', \mathsf{ID}_{A''}, \mathsf{ID}_B)$ for which $\mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$. We know that e is an encryption of $m_b||s_b||\mathsf{ID}_{A'}$. Since $\mathcal{R}_{\mathcal{E}}$ is decryption-respecting, e' too is an encryption of $m_b||s_b||\mathsf{ID}_{A'}$. Now there are two cases to consider. If $\mathsf{ID}_{A''} \neq \mathsf{ID}_{A'}$ then step 2 of VerDec would return $\perp$, just as step 1 above. Otherwise $\mathsf{ID}_{A''} = \mathsf{ID}_{A'}$, and then $\mathcal{R}_{\mathcal{S}t\mathcal{E}'}((e, \mathsf{ID}_{A'}, \mathsf{ID}_B), (e', \mathsf{ID}_{A''}, \mathsf{ID}_B)) = \mathcal{R}_{\mathcal{E}}(e, e') = \mathsf{true}$. But this contradicts the fact that $\mathcal{A}'_{\mathcal{S}t\mathcal{E}}$ does not query its de-signcryption oracle with anything equivalent to the target ciphertext.

Authenticity: Suppose there is an adversary $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ which can break the $\mathsf{UF\text{-}CMA}$ security of $\mathcal{E}t\mathcal{S}'$ with non-negligible probability. We convert this to an adversary $\mathcal{A}_{\mathcal{S}}$ which can break the $\mathsf{UF\text{-}CMA}$ security of $\mathcal{S}$ with the same probability, in the same time, and using the same number of oracle queries.

Let $(\mathsf{SK}_A, \mathsf{VK}_A)$ be the key pair which $\mathcal{A}_{\mathcal{S}}$ is trying to attack. After being given VK_A, $\mathcal{A}_{\mathcal{S}}$ does the following:

1. $(\mathsf{EK}_A, \mathsf{DK}_A) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
2. $(e, s, \mathsf{ID}_A, \mathsf{ID}_{B'}) \leftarrow \mathcal{A}_{\mathcal{E}t\mathcal{S}'}((\mathsf{VK}_A, \mathsf{EK}_A))$
3. Return $(e||\mathsf{ID}_{B'}, s)$.

In step 2, $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ may make queries to its signcryption oracle. To respond to a query $(m', \mathsf{ID}_A, \mathsf{ID}_{B''})$, $\mathcal{A}_{\mathcal{S}}$ does the following:

1. $e' \leftarrow \mathsf{Enc}_{\mathsf{EK}_{B''}}(m'||\mathsf{ID}_A)$
2. $s' \leftarrow \mathsf{Sig}_{\mathsf{SK}_A}(e'||\mathsf{ID}_{B''})$
3. Return (e', s').

Note that $\mathcal{A}_{\mathcal{S}}$ may perform step 2 using its signing oracle. We must check, however, that the message $e||\mathsf{ID}_{B'}$ signed by $\mathcal{A}_{\mathcal{S}}$ is not equal to any of the messages $e'||\mathsf{ID}_{B''}$ passed to its signing oracle. Suppose otherwise, i.e. $\mathsf{ID}_{B'} = \mathsf{ID}_{B''}$ and $e = e'$. Note that $e = \mathsf{Enc}_{\mathsf{EK}_{B'}}(m||\mathsf{ID}_A)$ for some m, and $e' = \mathsf{Enc}_{\mathsf{EK}_{B''}}(m'||\mathsf{ID}_A) = \mathsf{Enc}_{\mathsf{EK}_{B'}}(m'||\mathsf{ID}_A)$ for some m'. Then since $e = e'$ we have $\mathsf{Enc}_{\mathsf{EK}_{B'}}(m||\mathsf{ID}_A) = \mathsf{Enc}_{\mathsf{EK}_{B'}}(m'||\mathsf{ID}_A)$, which implies that $m||\mathsf{ID}_A = m'||\mathsf{ID}_A$, so $m = m'$, contradicting the fact that $\mathcal{A}_{\mathcal{E}t\mathcal{S}'}$ produces a signcryption for a message not queried to its signcryption oracle. $\square$

Strengthening These Results

As it turns out, we can improve upon these theorems.

Looking at the proof of Theorem 3.8, we see that if $\mathcal{R}_{\mathcal{E}}$ is the ordinary equality relation, then so is $\mathcal{R}_{\mathcal{S}t\mathcal{E}'}$. That is, if $\mathcal{E}$ is IND-CCA2-secure, then so is $\mathcal{S}t\mathcal{E}'$. In the case of $\mathcal{E}t\mathcal{S}'$, though, we cannot expect to do better than IND-gCCA2 security, at least in the insider model. To see this, note that the adversary can always de-signcrypt its target ciphertext by removing its signature, replacing it with a new one (which it can do, since it has access to SK_A), and then passing the "new" signcryption to its de-signcryption oracle.

Now we turn our attention to authenticity. Looking at the proof of Theorem 3.7, we see that UF-CMA security can be replaced by sUF-CMA-security. That is, if $\mathcal{S}$ is sUF-CMA-secure, then so is $\mathcal{E}t\mathcal{S}'$. This is not the case for $\mathcal{S}t\mathcal{E}$, at least in the insider model. The adversary can produce a new signcryption from an old one by decrypting it (using DK_B), then re-encrypting it.

$\mathcal{E}t\mathcal{S}'$ versus $\mathcal{S}t\mathcal{E}'$

Seeing that both sign-then-encrypt and encrypt-then-sign provide strong levels of security, one may wonder why the former is typically preferred over the latter. One reason is that sign-then-encrypt provides a very simple non-repudiation mechanism: the recipient of a signcrypted message simply decrypts it with his decryption key, then passes the resulting signed message on to the verifier. That is, $\mathcal{S}t\mathcal{E}'$ is an $\mathcal{S}$-verifiable signcryption scheme.

Encrypt-then-sign, as presented here, does not provide non-repudiation unless the recipient is prepared to divulge his secret key. Without it, a verifier cannot decrypt the signed ciphertext. This problem can be solved if the recipient can determine the randomness r used by the sender while encrypting. (For example, we could required the sender to append it to the message m before encrypting.) In this case, the recipient can provide m, r, and the signature s to the verifier, who can then encrypt m under the recipient's public key using r, and verify the signature s on the resulting ciphertext.

3.5 Encrypt-and-Sign

Another method to achieve both confidentiality and authenticity, which also existed before the study of signcryption, is *encrypt-and-sign*, or $\mathcal{E}\&\mathcal{S}$. Rather than applying the encryption and signature operations sequentially,

as with $\mathcal{E}t\mathcal{S}$, we apply them in parallel. For simplicity, we present the definition of encrypt-and-sign in the two-user setting.

Definition 3.9 (Encrypt-and-sign). *Given an encryption scheme $\mathcal{E}$ and a signature scheme $\mathcal{S}$, we define the signcryption scheme $\mathcal{E}\&\mathcal{S}$ as follows:*

- $\mathsf{KeyGen}(1^k)$*:*
  1. $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
  2. $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
  3. *Return* $(\mathsf{SDK}, \mathsf{VEK}) \leftarrow ((\mathsf{SK}, \mathsf{DK}), (\mathsf{VK}, \mathsf{EK}))$.
- $\mathsf{SigEnc}_{\mathsf{SDK}_A, \mathsf{VEK}_B}(m)$*:*
  1. *Return* $(e, s) \leftarrow (\mathsf{Enc}_{\mathsf{EK}_B}(m), \mathsf{Sig}_{\mathsf{SK}_A}(m))$
- $\mathsf{VerDec}_{\mathsf{VEK}_A, \mathsf{SDK}_B}(e, s)$*:*
  1. $m \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(e)$
  2. *If* $\mathsf{Ver}_{\mathsf{VK}_A}(m, s) = \mathsf{succeed}$ *return* m *else return* $\perp$.

Note that encrypt-and-sign requires the same amount of computation as encrypt-then-sign, but it allows the encryption and signature steps to be performed in parallel. The decryption and verification steps, however, must be performed sequentially.

The main problem with encrypt-and-sign is that it does not provide indistinguishability. This is because an adversary can check whether a given message m corresponds to a given ciphertext (e, s) by running $\mathsf{Ver}_{\mathsf{VK}_A}(m, s)$. Encrypt-and-sign does, however, provide non-repudiation. The recipient of a message (e, s) simply computes $m \leftarrow \mathsf{Dec}_{\mathsf{DK}_B}(e)$ and then sends (m, s) to the verifier.

3.6 Improvements on Encrypt-and-Sign

An, Dodis, and Rabin [1] solved the problem of indistinguishability in $\mathcal{E}\&\mathcal{S}$ by adding a commitment step, resulting in *commit-then-encrypt-and-sign* or $\mathcal{C}t\mathcal{E}\&\mathcal{S}$. Besides providing indistinguishability, $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ also improves upon $\mathcal{E}\&\mathcal{S}$ in that it allows the decryption and verification steps to be performed in parallel.

Pieprzyk and Pointcheval [12] subsequently improved upon $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ by replacing the commitment portion with a Shamir secret-sharing scheme. Their scheme has the added advantage that the encryption and signature schemes it depends on need only satisfy very weak security requirements.

We do not consider these schemes in detail, and instead move on to other classes of signcryption schemes.

Chapter 4

Signcryption in the Discrete Logarithm Setting

In this chapter, we examine a class of signcryption schemes which are built from signature schemes in the discrete logarithm setting.

4.1 Notation

Before describing particular schemes, we fix some notation which is common to all of them.

- (p, q, g) — parameters defining a multiplicative subgroup of $\mathbb{Z}_p$, with order q and generator g. That is, p is a large prime, q is a large prime factor of $p - 1$, and $g \in [1, \ldots, p - 1]$ has order q.

  In practice p and q are chosen so that solving the discrete logarithm problem using the index calculus algorithm in $\mathbb{Z}_p$ takes about the same amount of time as solving it in the subgroup generated by g using Pollard's rho algorithm. Typical sizes for p and q are 1024 and 160 bits respectively.

  In encryption and signature schemes, these values can typically be chosen independently by each user. However, in the signcryption schemes in this chapter, (p, q, g) are common to all users.

- (x_A, y_A) and (x_B, y_B) — the key pairs of the sender and recipient respectively. These key pairs are generated using the following key generation algorithm, which is common to all signature and signcryp-

tion schemes in this chapter. A user U generates his key pair (x_U, y_U) by doing:

1. $x_U \leftarrow^R [1, \ldots, q-1]$
2. $y_U \leftarrow g^{x_U} \bmod p$.

The value x_U is the private key, while y_U is the public key.

Since this key generation algorithm is common to all schemes in this chapter, we specify only Sig and Ver when describing signature schemes, and SigEnc and VerDec when describing signcryption schemes. We take both $\mathsf{Sig\text{-}KeyGen}(1^k)$ and $\mathsf{KeyGen}(1^k)$ to be the above algorithm, with the security parameter k being the length (in bits) of p.

- (E, D) — the encryption and decryption algorithms of a private-key encryption scheme. This scheme must be deterministic and secure against chosen plaintext attacks. For example, triple-DES or IDEA could be used in the CBC mode of operation.

- G, H — hash functions, each of which maps from $\{0,1\}^*$ to $\{0,1\}^l$ for some l. (The length parameter l need not be the same for G and H.) They are assumed to behave like random oracles[1] for the purpose of security proofs.

4.2 Zheng's Original Scheme

In 1997, Zheng effectively launched the study of signcryption by giving a pair of signcryption schemes which are significantly more efficient than $\mathcal{StE}$ both in terms of computation and message expansion [15]. These schemes have served as a template for several other signcryption schemes, including those presented in this chapter.

Here we present the first of Zheng's signcryption schemes. It is based on a shortened version of the Digital Signature Standard known as SDSS1 (Shortened Digital Signature Standard). This signature scheme is defined as follows:

- $\mathsf{Sig}(m)$:

[1] A *random oracle* is a theoretical model of a perfect cryptographic hash function. Given a new input $x \in \{0,1\}^*$, it chooses an output $y \in \{0,1\}^l$ uniformly at random. Given a previously seen input, it returns the same output it previously chose. For more information, see Bellare and Rogaway [4].

1. $x \leftarrow^R [1, \ldots, q-1]$
2. $k \leftarrow g^x \bmod p$
3. $r \leftarrow \mathsf{H}(k||m)$
4. $s \leftarrow x/(r + x_A) \bmod q$
5. Return (m, r, s).

- $\mathsf{Ver}(m, r, s)$:
  1. $k \leftarrow (y_A \cdot g^r)^s \bmod p$
  2. If $r = \mathsf{H}(k||m)$ return succeed else return fail.

This particular signature scheme was chosen by Zheng because the value k can be computed by the recipient without knowing the message m.

Now we can define Zheng's signcryption scheme[2]:

- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $x \leftarrow^R [1, \ldots, q-1]$
  2. $K \leftarrow y_B^x \bmod p$
  3. $K_{enc} \leftarrow \mathsf{G}(K)$
  4. $c \leftarrow \mathsf{E}_{K_{enc}}(m)$
  5. $r \leftarrow \mathsf{H}(m||y_A||y_B||K)$
  6. $s \leftarrow x/(r + x_A) \bmod q$
  7. Return (c, r, s).
- $\mathsf{VerDec}(c, r, s, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $K \leftarrow (y_A \cdot g^r)^{s \cdot x_B} \bmod p$
  2. $K_{enc} \leftarrow \mathsf{G}(K)$
  3. $m \leftarrow \mathsf{D}_{K_{enc}}(c)$
  4. If $r = \mathsf{H}(m||y_A||y_B||K)$ return m, else return $\perp$.

This scheme is the result of making the following modifications to SDSS1:

1. In the signcryption algorithm, the computation $k \leftarrow g^x \bmod p$ is replaced by $K \leftarrow y_B^x \bmod p$, and the corresponding change is made in the de-signcryption algorithm. This change ensures that only the intended recipient can compute K.

2. The message m is hidden by encrypting it, using $\mathsf{G}(K)$ as the symmetric encryption key.

[2]This version of Zheng's scheme is not the one originally presented by Zheng [15], but a slightly modified version used by Baek, Steinfeld, and Zheng [2] in their formal security proof.

3. The computation $r \leftarrow \mathsf{H}(k||m)$ is replaced by $r \leftarrow \mathsf{H}(m||y_A||y_B||K)$, which binds the signcryption to the identities of the sender and recipient, in order to achieve security in the multi-user setting.

The main advantage of Zheng's scheme is that it requires significantly less computation than traditional signcryption techniques such as $\mathcal{S}t\mathcal{E}$. To see this, note that Zheng's scheme requires only one modular exponentiation for signcryption, and two modular exponentiations for de-signcryption. In comparison, $\mathcal{S}t\mathcal{E}$ requires three modular exponentiations both for signcryption and for de-signcryption, when using Schnorr signatures and ElGamal encryption.

Zheng's scheme also has an advantage in terms of communication overhead, expanding messages by only $|\mathsf{H}(\cdot)| + |q|$ bits, compared to $|\mathsf{H}(\cdot)| + |q| + |p|$ in the case of $\mathcal{S}t\mathcal{E}$ with Schnorr signatures and ElGamal encryption. This difference is quite significant, since $|p|$ is typically much larger than $|\mathsf{H}(\cdot)|$ and $|q|$. It becomes even more significant when Zheng's scheme is compared with $\mathcal{S}t\mathcal{E}'$, which expands messages by $|\mathsf{H}(\cdot)| + |q| + |p| + |\mathsf{ID}_A|$ bits, with the extra $|\mathsf{ID}_A|$ coming from the fact that ID_A is appended before the encryption step. ID_A is typically around the same length as $\mathsf{H}(\cdot)$.

Security

Zheng's scheme provides past recovery, since the sender of a message (c, r, s) can compute $x = s(r + x_A) \bmod q$, and thus $K = y_B^x \bmod p$, and then perform the remaining steps of VerDec as in the definition. Of course, this implies that the scheme does not provide confidentiality in the insider security model.

Baek, Steinfeld, and Zheng [2] have given a formal security proof for this scheme. Before proceeding to their results, we briefly look at the hard problems upon which they are based. They are as follows:

- **Computational Diffie-Hellman Problem (CDHP)**: For $a, b \in \mathbb{Z}_q^*$, given g, g^a, and g^b, compute g^{ab}.
- **Decisional Diffie-Hellman Problem (DDHP)**: For $a, b, c \in \mathbb{Z}_q^*$, given g, g^a, g^b, and g^c, decide whether $c \equiv ab \pmod q$.
- **Gap Diffie-Hellman Problem (GDHP)**: Solve CDHP, given an oracle to solve DDHP.

Note that the computational and decisional Diffie-Hellman problems have been well studied, while less is known about the more recent gap Diffie-Hellman problem, thus casting some doubt on security proofs based upon

it. Further details about the gap Diffie-Hellman problem can be found in Okamoto and Pointcheval [9].

Now we return to the results of Baek et al. They prove that Zheng's scheme provides confidentiality in a sense slightly weaker than IND-CCA2-security in the outsider model, under the assumption that the gap Diffie-Hellman problem is hard. They show that it provides authenticity in a sense slightly weaker than UF-CMA-security in the insider model, under the assumption that the computational Diffie-Hellman problem is hard. In both cases, the difference between their model and the one presented here is that the signcryption oracle is $\mathsf{SigEnc}(\cdot, \mathsf{ID}_A, \mathsf{ID}_B)$, rather than $\mathsf{SigEnc}(\cdot, \mathsf{ID}_A, \cdot)$. That is, the adversary does not get to choose the recipient when asking for the signcryption of a message. (The de-signcryption oracle, however, is left unchanged.)

4.3 Adding Non-Repudiation

Zheng's original scheme did provide a mechanism for non-repudiation. However, Petersen and Michels [11] showed that this mechanism compromises the confidentiality of the system. To overcome this problem, Bao and Deng [3] modified Zheng's scheme, giving the following signcryption scheme[3]:

- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  * 1. $x \leftarrow^R [1, \ldots, q-1]$
  * 2. $k \leftarrow g^x \bmod p$
  3. $K \leftarrow y_B^x \bmod p$
  4. $K_{enc} \leftarrow \mathsf{G}(K)$
  5. $c \leftarrow \mathsf{E}_{K_{enc}}(m)$
  * 6. $r \leftarrow \mathsf{H}(k||m||y_A||y_B)$
  * 7. $s \leftarrow x/(r + x_A) \bmod q$
  8. Return (c, r, s).
- $\mathsf{VerDec}(c, r, s, \mathsf{ID}_A, \mathsf{ID}_B)$:
  * 1. $k \leftarrow (y_A \cdot g^r)^s \bmod p$
  2. $K \leftarrow k^{x_B} \bmod p$
  3. $K_{enc} \leftarrow \mathsf{G}(K)$
  4. $m \leftarrow \mathsf{D}_{K_{enc}}(c)$
  * 5. If $r = \mathsf{H}(k||m||y_A||y_B)$ return m, else return $\perp$.

[3] In fact, Bao and Deng specified $r \leftarrow \mathsf{H}(m||y_A||y_B||k)$ in step 6, but we move k to the beginning here to allow for SDSS1-verifiability.

This scheme is very similar to Zheng's, except for the following changes:

1. The computations $k \leftarrow g^x \bmod p$ and $k \leftarrow (y_A \cdot g^r)^s \bmod p$ from the original SDSS1 signature scheme are added back.

2. The computation $r \leftarrow \mathsf{H}(m||y_A||y_B||K)$ is replaced by $r \leftarrow \mathsf{H}(k||m||y_A||y_B)$.

Thanks to these changes, the steps of SigEnc with asterisks (*) are precisely the signature algorithm of SDSS1, when signing the message $m||y_A||y_B$. Similarly, the steps of VerDec with asterisks are precisely the verification algorithm of SDSS1, when verifying the same message. Thus to achieve non-repudiation, the recipient of a message $(c, r, s, \mathsf{ID}_A, \mathsf{ID}_B)$ can simply compute m as in VerDec, and then send $(m||y_A||y_B, r, s)$ to the verifier, who then verifies that this is a valid SDSS1 signature using A's public key. Thus this is an SDSS1-verifiable signature scheme.

In this scheme, signcryption requires two modular exponentiations, while de-signcryption requires three. Thus most of the computational advantage of Zheng's scheme over $\mathcal{S}t\mathcal{E}$ has been lost. However, the advantage in communication overhead remains, since the overhead of Bao and Deng's scheme is $|\mathsf{H}(\cdot)| + |q|$ bits, as with Zheng's scheme.

Note that this scheme provides insider security against UF-CMA attacks as long as SDSS1 is UF-CMA-secure[4]. To see this, suppose there exists an adversary $\mathcal{A}$ which can break its UF-CMA-security with non-negligible probability. Then we can construct an adversary $\mathcal{A}'$ which breaks the UF-CMA-security of SDSS1 with the same probability and in the same time. After receiving a public key y_A, $\mathcal{A}'$ does:

1. $(c, r, s, \mathsf{ID}_{B'}) \leftarrow \mathcal{A}(y_A)$
2. $m \leftarrow \mathsf{VerDec}(c, r, s, \mathsf{ID}_A, \mathsf{ID}_{B'})$
3. Return $(m||y_A||y_{B'}, r, s)$.

Note that $\mathcal{A}'$ can perform step 2 since it has access to $x_{B'}$. (We assume that whatever key pairs and identities are generated by $\mathcal{A}$ are made available to $\mathcal{A}'$ as well.)

Unfortunately, this scheme does not provide IND-CCA2 security, since an adversary can easily determine which of two messages m_0, m_1 corresponds to a challenge ciphertext $(c, r, s, \mathsf{ID}_A, \mathsf{ID}_B)$ checking which of $(m_0||y_A||y_B, r, s)$ or $(m_1||y_A||y_B, r, s)$ is a valid SDSS1 signature.

[4]Zheng [15] claims that SDSS1 can be proven UF-CMA-secure by adapting the argument of Pointcheval and Stern [13] for the Schnorr signature scheme.

4.4 A DSA-Verifiable Scheme

Shin, Lee, and Shim [14] observed that Bao and Deng's method could be used to construct signcryption schemes from an entire class of ElGamal-type signature schemes; namely, those for which $k = g^x \bmod p$ can be computed by the recipient of a signature, without knowing the message m. Suppose that $\mathcal{S}$ is such a signature scheme. Then the corresponding $\mathcal{S}$-verifiable signcryption scheme is:

- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $x \leftarrow^R [1, \ldots, q-1]$
  2. $k \leftarrow g^x \bmod p$
  3. $K \leftarrow y_B^x \bmod p$
  4. $K_{enc} \leftarrow \mathsf{G}(K)$
  5. $c \leftarrow \mathsf{E}_{K_{enc}}(m)$
  6. Compute A's signature $s \leftarrow \mathsf{Sig}(m||y_A||y_B)$ using the values x and k from above.
  7. Return (c, s).
- $\mathsf{VerDec}(c, s, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. Compute $k = g^x \bmod p$ as specified by $\mathcal{S}$.
  2. $K \leftarrow k^{x_B} \bmod p$
  3. $K_{enc} \leftarrow \mathsf{G}(K)$
  4. $m \leftarrow \mathsf{D}_{K_{enc}}(c)$
  5. Compute $a \leftarrow \mathsf{Ver}(m||y_A||y_B, s)$ using the value $k = g^x \bmod p$ computed in step 1.
  6. If $a = \mathsf{succeed}$ return m, otherwise return $\perp$.

To achieve non-repudiation, the recipient of a message (c, s) simply computes m as in VerDec, then sends $(m||y_A||y_B, s)$ to the verifier, who then verifies that this is a valid signature using $\mathcal{S}$.

This indeed generalizes Bao and Deng's scheme, since their scheme results from applying the above construction to SDSS1.

One would like to apply this construction to the widely-used Digital Signature Algorithm (DSA), which is defined as follows:

- $\mathsf{Sig}(m)$:
  1. $x \leftarrow^R [1, \ldots, q-1]$
  2. $k \leftarrow g^x \bmod p$
  3. $r \leftarrow k \bmod q$

4. $h \leftarrow \mathsf{H}(m)$
5. $s \leftarrow (h + x_A \cdot r)/x \bmod q$
6. Return (m, r, s).

- $\mathsf{Ver}(m, r, s)$:

1. $h \leftarrow \mathsf{H}(m)$
2. $e_1 \leftarrow h/s \bmod q$
3. $e_2 \leftarrow r/s \bmod q$
4. $k \leftarrow g^{e_1} \cdot y_A^{e_2} \bmod p$
5. If $r = k \bmod q$ return succeed, else return fail.

Unfortunately, DSA does not satisfy the requirement that $k = g^x \bmod p$ can be computed without knowing the message m, so the construction cannot be directly applied to DSA. However, Shin, Lee, and Shim were able to overcome this problem by introducing a modified version of DSA called MDSA:

- $\mathsf{Sig}(m)$:

1. $x \leftarrow^R [1, \ldots, q-1]$
2. $k \leftarrow g^x \bmod p$
3. $r \leftarrow k \bmod q$
4. $h \leftarrow \mathsf{H}(m)$
5. $s \leftarrow (h + x_A \cdot r)/x \bmod q$
6. $e_1 \leftarrow h/s \bmod q$
7. $e_2 \leftarrow r/s \bmod q$
8. Return (m, e_1, e_2).

- $\mathsf{Ver}(m, e_1, e_2)$:

1. $h \leftarrow \mathsf{H}(m)$
2. $k \leftarrow g^{e_1} \cdot y_A^{e_2} \bmod p$
3. $r \leftarrow k \bmod q$
4. $s \leftarrow r/e_2 \bmod q$
5. If $e_1 \cdot s \equiv h \bmod q$ return succeed, else return fail.

MDSA is essentially the same as DSA, except that the computation of e_1 and e_2 has been moved from Ver to Sig, and Ver has been adjusted accordingly.

Note that in step 2 the verifier computes

$$\begin{aligned} k &= g^{e_1} \cdot y_A^{e_2} \bmod p \\ &= g^{h/s} \cdot g^{x_A \cdot (r/s)} \bmod p \\ &= g^{(h + x_A \cdot r)/s} \bmod p \\ &= g^{(h + x_A \cdot r)/((h + x_A \cdot r)/x)} \bmod p \\ &= g^x \bmod p \end{aligned}$$

without using the message m. Thus the generalized construction of Bao and Deng can be applied to MDSA.

Further, MDSA is *strongly equivalent* to DSA. By this, we mean that a valid MDSA signature can be efficiently converted to a valid DSA signature and vice versa, without using the private key. The following is a restatement of the theorem and proof given by Shin, Lee, and Shim [14], with missing details filled in by the author.

Theorem 4.1. *MDSA is strongly equivalent to DSA.*

Proof. Suppose that (m, e_1, e_2) is a valid MDSA signature. That is, $e_1 \cdot s \equiv h \pmod q$, where $s = ((g^{e_1} \cdot y_A^{e_2} \bmod p) \bmod q)/e_2 \bmod q$ and $h = \mathsf{H}(m)$. Then (m, α, β) is a valid DSA signature, where $\alpha = (g^{e_1} \cdot y_A^{e_2} \bmod p) \bmod q$ and $\beta = \alpha/e_2 \bmod q$, since

$$\begin{aligned} e_1 \cdot s &\equiv h \pmod q \\ e_1 \cdot (\alpha/e_2) &\equiv \mathsf{H}(m) \pmod q \\ e_1 \cdot \beta &\equiv \mathsf{H}(m) \pmod q \\ e_1 &\equiv \mathsf{H}(m)/\beta \pmod q \\ g^{e_1} &\equiv g^{\mathsf{H}(m)/\beta} \pmod p \\ g^{e_1} \cdot y_A^{e_2} &\equiv g^{\mathsf{H}(m)/\beta} \cdot y_A^{e_2} \pmod p \\ g^{e_1} \cdot y_A^{e_2} &\equiv g^{\mathsf{H}(m)/\beta} \cdot y_A^{\alpha/\beta} \pmod p \\ \alpha = (g^{e_1} \cdot y_A^{e_2} \bmod p) \bmod q &= (g^{\mathsf{H}(m)/\beta} \cdot y_A^{\alpha/\beta} \bmod p) \bmod q. \end{aligned}$$

This last condition is precisely what DSA's verification algorithm checks to determine whether (m, α, β) is a valid signature.

Now suppose that (m, r, s) is a valid DSA signature. That is, $r = (g^{e_1} \cdot y_A^{e_2} \bmod p) \bmod q$, where $e_1 = \mathsf{H}(m)/s \bmod q$ and $e_2 = r/s \bmod q$. Then (m, α, β) is a valid MDSA signature, where $\alpha = \mathsf{H}(m)/s \bmod q$ and $\beta =$

$r/s \bmod q$, since

$$\begin{aligned}
r &\equiv (g^{e_1} \cdot y_A^{e_2} \bmod p) \pmod q \\
r &\equiv (g^{\alpha} \cdot y_A^{\beta} \bmod p) \pmod q \\
(\alpha/\beta) \cdot r &\equiv (\alpha/\beta) \cdot (g^{\alpha} \cdot y_A^{\beta} \bmod p) \pmod q \\
\left(\frac{\mathsf{H}(m)/s}{r/s}\right) \cdot r &\equiv \alpha \cdot ((g^{\alpha} \cdot y_A^{\beta} \bmod p)/\beta) \pmod q \\
\mathsf{H}(m) &= \alpha \cdot (((g^{\alpha} \cdot y_A^{\beta} \bmod p) \bmod q)/\beta \bmod q) \pmod q.
\end{aligned}$$

This last condition is precisely what MDSA's verification algorithm checks to determine whether (m, α, β) is a valid signature. □

It is easy to see that if two signature schemes are strongly equivalent, then they have the same security properties. Thus if DSA is UF-CMA-secure, then so is MDSA. At present, it has not been proven that DSA is UF-CMA-secure, but it is widely used in practise, and no significant security flaws are known.

By applying the generalized technique of Bao and Deng to MDSA, we get an MDSA-verifiable signcryption scheme. Then using the strong equivalence of MDSA and DSA, we can easily modify the non-repudiation procedure to get the following DSA-verifiable signcryption scheme:

- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $x \leftarrow^R [1, \ldots, q-1]$
  2. $k \leftarrow g^x \bmod p$
  3. $K \leftarrow y_B^x \bmod p$
  4. $K_{enc} \leftarrow \mathsf{G}(K)$
  5. $c \leftarrow \mathsf{E}_{K_{enc}}(m)$
  6. $r \leftarrow k \bmod q$
  7. $h \leftarrow \mathsf{H}(m||y_A||y_B)$
  8. $s \leftarrow (h + x_A \cdot r)/x \bmod q$
  9. $e_1 \leftarrow h/s \bmod q$
  10. $e_2 \leftarrow r/s \bmod q$
  11. Return (c, e_1, e_2).
- $\mathsf{VerDec}(c, e_1, e_2, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $k \leftarrow g^{e_1} \cdot y_A^{e_2} \bmod p$
  2. $K \leftarrow k^{x_B} \bmod p$
  3. $K_{enc} \leftarrow \mathsf{G}(K)$

4. $m \leftarrow \mathsf{D}_{K_{enc}}(c)$
5. $h \leftarrow \mathsf{H}(m||y_A||y_B)$
6. $r \leftarrow k \bmod q$
7. $s \leftarrow r/e_2 \bmod q$
8. If $e_1 \cdot s \equiv h \bmod q$ return m, else return $\perp$.

To achieve non-repudiation, the recipient sends $(m||y_A||y_B, r, s)$ to the verifier, where m, r, and s are the values computed in VerDec. If (c, e_1, e_2, ID_A, ID_B) is a valid signcryption, then $(m||y_A||y_B, e_1, e_2)$ is a valid MDSA signature, and thus $(m||y_A||y_B, r, s)$ is a valid DSA signature, as shown in Theorem 4.1.

Unfortunately, this scheme does not provide IND-CCA2 security. As with Bao and Deng's scheme, an adversary can easily determine which of two messages m_0, m_1 corresponds to a challenge ciphertext $(c, e_1, e_2, \mathsf{ID}_A, \mathsf{ID}_B)$ checking which of $(m_0||y_A||y_B, e_1, e_2)$ or $(m_1||y_A||y_B, e_1, e_2)$ is a valid MDSA signature. To solve this problem, Shin, Lee, and Shim propose a slightly modified version of the scheme called SC-DSA+. It is defined as follows:

- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $x \leftarrow^R [1, \ldots, q-1]$
  2. $k \leftarrow g^x \bmod p$
  3. $K \leftarrow y_B^x \bmod p$
  4. $K_{enc} \leftarrow \mathsf{G}(K)$
  5. $c \leftarrow \mathsf{E}_{K_{enc}}(m)$
  6. $r \leftarrow k \bmod q$
  7. $h \leftarrow \mathsf{H}(m||y_A||y_B||K)$
  8. $s \leftarrow (h + x_A \cdot r)/x \bmod q$
  9. $e_1 \leftarrow h/s \bmod q$
  10. $e_2 \leftarrow r/s \bmod q$
  11. Return (c, e_1, e_2).
- $\mathsf{VerDec}(c, e_1, e_2, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $k \leftarrow g^{e_1} \cdot y_A^{e_2} \bmod p$
  2. $K \leftarrow k^{x_B} \bmod p$
  3. $K_{enc} \leftarrow \mathsf{G}(K)$
  4. $m \leftarrow \mathsf{D}_{K_{enc}}(c)$
  5. $r \leftarrow k \bmod q$
  6. $s \leftarrow r/e_2 \bmod q$
  7. $h \leftarrow \mathsf{H}(m||y_A||y_B||K)$

8. If $e_1 \cdot s \equiv h \bmod q$ return m else return $\perp$.

To achieve non-repudiation, the recipient of a message simply computes m, r, s, and K as in VerDec, and sends $(m||y_A||y_B||K, r, s)$ to the verifier, who then verifies that this is a valid DSA signature using A's public key.

This scheme is precisely the same as the previous one, except that the step $h \leftarrow \mathsf{H}(m||y_A||y_B)$ has been changed to $h \leftarrow \mathsf{H}(m||y_A||y_B||K)$. This prevents the previous simple attack on $\mathsf{IND\text{-}CCA2}$ security, since an outsider adversary cannot efficiently compute the value K.

As with Bao and Deng's scheme in Section 4.3, it is straightforward that SC-DSA+ is $\mathsf{UF\text{-}CMA}$-secure in the insider model as long as MDSA (or equivalently, DSA) is $\mathsf{UF\text{-}CMA}$-secure. The proof of confidentiality for SC-DSA+ is simply an adaptation of the proof for Zheng's original scheme, and uses the same security model. That is, SC-DSA+ provides confidentiality in a sense slightly weaker than $\mathsf{IND\text{-}CCA2}$-security in the outsider model.

Note that this scheme provides past recovery, since the sender of a message $(c, e_1, e_2, \mathsf{ID}_A, \mathsf{ID}_B)$ can compute:

1. $k \leftarrow g^{e_1} \cdot y_A^{e_2} \bmod p$
2. $r \leftarrow k \bmod q$
3. $s \leftarrow r/e_2 \bmod q$
4. $h \leftarrow e_1 \cdot s \bmod q$
5. $x \leftarrow (h + x_A \cdot r)/s \bmod q$
6. $K \leftarrow y_B^x \bmod p$
7. $K_{enc} \leftarrow \mathsf{G}(K)$
8. $m \leftarrow \mathsf{D}_{K_{enc}}(c)$.

Chapter 5

Signcryption from Trapdoor Permutations

5.1 Trapdoor Permutation Families

Trapdoor permutations are an important building block of public-key cryptography. A trapdoor permutation is simply a permutation $f : S \to S$ on some finite set S, which can be efficiently evaluated by anyone, but whose inverse permutation $f^{-1} : S \to S$ can only be efficiently evaluated by using some secret trapdoor information.

5.1.1 Syntax

More formally, we define a trapdoor permutation family as a triple of algorithms $(\mathsf{Trap\text{-}Gen}, \mathsf{Eval}, \mathsf{Invert})$:

- $\mathsf{Trap\text{-}Gen}$ is a randomized algorithm which accepts 1^k, where k is a security parameter, and outputs a pair (f, f^{-1}), where f is a permutation over some set S and f^{-1} is its inverse permutation. This operation is denoted by $(f, f^{-1}) \leftarrow \mathsf{Trap\text{-}Gen}(1^k)$.

- Eval is a deterministic algorithm which accepts a permutation f generated by $\mathsf{Trap\text{-}Gen}$, as well as some $x \in S$, where S is the set over which f is defined. It outputs some $y \in S$. This operation is denoted by $y \leftarrow \mathsf{Eval}(f, x)$, or simply $y \leftarrow f(x)$.

- Invert is a deterministic algorithm which accepts some f^{-1} generated by $\mathsf{Trap\text{-}Gen}$ and some $y \in S$, where S is the set over which f is

defined. It outputs some $x \in S$. This operation is denoted by $x \leftarrow \mathsf{Invert}(f^{-1}, y)$, or simply $x \leftarrow f^{-1}(y)$.

We require that $f^{-1}(f(x)) = x$ for all x, and for all pairs (f, f^{-1}) generated by Trap-Gen. This ensures both that f is a permutation, and that f^{-1} is the inverse permutation of f.

5.1.2 Security

As was the case for encryption, signature, and signcryption schemes, we define the security of a trapdoor permutation family in terms of the goal and capabilities of an adversary.

We will take the goal of the adversary to be inverting a randomly chosen element. To achieve this goal, the adversary $\mathcal{A}$ must win the following game:

1. $(f, f^{-1}) \leftarrow \mathsf{Trap\text{-}Gen}(1^k)$
2. $y \leftarrow^R S$ (where S is the set over which f is defined)
3. $x \leftarrow \mathcal{A}(f, y)$.

$\mathcal{A}$ wins in the case that $x = f^{-1}(y)$. As for the capabilities of the adversary, we only give it access to the permutation f. We require that a trapdoor permutation family satisfy $\Pr[\mathcal{A} \text{ wins}] < \mathsf{negl}(k)$ for any such adversary $\mathcal{A}$.

Note that we have chosen a particularly strong goal for the adversary, and given it particularly weak capabilities. (A weaker goal would, for example, be one which required $\mathcal{A}$ to determine only *some* information about x from y. Stronger capabilities could include access to an oracle for f^{-1}.) This means that the security level provided by a trapdoor permutation family is quite weak. However, it is still possible to build strongly secure encryption and signature schemes from trapdoor permutation families. As we will see in this chapter, it is also possible to build strongly secure signcryption schemes from them. Thus their weak notion of security is an advantage, making them easier to construct and analyze.

5.1.3 Examples

The most commonly used trapdoor permutation family is RSA, which is defined as follows:

- Trap-Gen(1^k):
  1. Choose two random k-bit primes, p and q.
  2. $n \leftarrow p \cdot q$

3. $\phi(n) \leftarrow (p-1) \cdot (q-1)$
4. Choose e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$.
5. Compute d such that $1 < d < \phi(n)$ and $d \cdot e \equiv 1 \pmod{\phi(n)}$.
6. Return $((e, n), (d, n))$.

- $\mathsf{Eval}((e, n), x)$:
  1. Return $y \leftarrow x^e \bmod n$.
- $\mathsf{Invert}((d, n), y)$:
  1. Return $x \leftarrow y^d \bmod n$.

Note that it is not yet known whether RSA is in fact a trapdoor permutation family, because it has not been proven to satisfy the security requirement defined above, even under the well-accepted assumption that factoring n is intractable. However, there are other families which have been proven secure, such as that of Paillier [10], which has been shown to be secure under the assumption that factoring is hard.

5.2 Cryptography from Trapdoor Permutations

Originally, trapdoor permutations were used directly as encryption and signature schemes. For example, to encrypt a message m for a recipient B, one would simply compute $c = f_B(m)$, where f_B is B's trapdoor function. Of course, the recipient would then compute $m = f_B^{-1}(c)$ using his secret trapdoor information. Unfortunately, this simple scheme does not provide indistinguishability, since anyone can distinguish between $f(m_0)$ and $f(m_1)$, for any m_0 and m_1. Similarly, to sign a message m, a sender A would simply compute $s = f_A^{-1}(m)$, and then anyone could verify that $f_A(s) = m$. However, this does not provide unforgeability, since any user can compute valid message/signature pairs $(f_A(s'), s')$ for any s'.

The solution to both of these problems is to first apply a padding function to the message, then apply the trapdoor permutation to some or all of the padded message. Thus a trapdoor-based encryption scheme typically looks like:

- $\mathsf{Enc\text{-}KeyGen}(1^k)$:
  1. $(f, f^{-1}) \leftarrow \mathsf{Trap\text{-}Gen}(1^k)$
  2. Return (f, f^{-1}).
- $\mathsf{Enc}(m)$:
  1. $m' \leftarrow \mathsf{Enc\text{-}Pad}(m)$

2. Return $c \leftarrow f_B(m')$.

- $\mathsf{Dec}(c)$:
  1. $m' \leftarrow f_B^{-1}(c)$
  2. Return $m \leftarrow \mathsf{Enc\text{-}Depad}(m')$.

Likewise, a trapdoor-based signature scheme typically looks like:

- $\mathsf{KeyGen}(1^k)$:
  1. $(g, g^{-1}) \leftarrow \mathsf{Trap\text{-}Gen}(1^k)$
  2. Return (g^{-1}, g).
- $\mathsf{Sig}(m)$:
  1. $m' \leftarrow \mathsf{Sig\text{-}Pad}(m)$
  2. $s \leftarrow g_A^{-1}(m')$
  3. Return (m, s).
- $\mathsf{Ver}(m, s)$:
  1. $m' \leftarrow \mathsf{Sig\text{-}Pad}(m)$
  2. $m'' \leftarrow g_A(s)$
  3. If $m' = m''$ return $\mathsf{succeed}$ else return fail.

Since there are many such encryption and signature schemes based on trapdoor permutations, for example OAEP and PSS-R, it would be possible to build signcryption schemes from them using the generic $\mathcal{S}t\mathcal{E}'$ and $\mathcal{E}t\mathcal{S}'$ constructions from Chapter 3. However, this approach has several drawbacks. To see this, let us consider the key generation and signcryption functions in the case of $\mathcal{S}t\mathcal{E}'$:

- $\mathsf{KeyGen}(1^k)$:
  1. $(f, f^{-1}) \leftarrow \mathsf{Enc\text{-}KeyGen}(1^k)$
  2. $(g^{-1}, g) \leftarrow \mathsf{Sig\text{-}KeyGen}(1^k)$
  3. Return $((g^{-1}, f^{-1}), (g, f))$.
- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $m' \leftarrow \mathsf{Sig\text{-}Pad}(m||\mathsf{ID}_B)$
  2. $s \leftarrow g_A^{-1}(m')$
  3. $s' \leftarrow \mathsf{Enc\text{-}Pad}(m||s||\mathsf{ID}_A)$
  4. $u \leftarrow f_B(s')$
  5. Return $(u, \mathsf{ID}_A, \mathsf{ID}_B)$.

This scheme has the following drawbacks:

- Each user must maintain two distinct trapdoor permutations: one for encrypting (f), and one for signing (g).
- Two padding steps are required; one for encryption, and one for signature. Since padding steps generally lengthen their inputs, this unnecessarily lengthens the input to the second trapdoor permutation, as well as the final ciphertext output.
- The identity of the recipient is appended before signing, and the identity of the sender is appended before encrypting. This unnecessarily lengthens the inputs to the trapdoor permutations, as well as the final ciphertext output.

Dodis, Freedman, Jarecki, and Walfish [5] have proposed a series of trapdoor-based signcryption schemes which addresses all three of these problems. Their schemes require each user to maintain only a single trapdoor permutation, which is used to achieve both confidentiality and authenticity. Only a single padding step is required, which is applied just before the trapdoor permutations of the sender and recipient. The identities of the sender and recipient are fed in as inputs to the padding step, but do not lengthen its output. Abstractly, their schemes all have the following pattern:

- $\mathsf{KeyGen}(1^k)$:
  1. $(f, f^{-1}) \leftarrow \mathsf{Trap\text{-}Gen}(1^k)$
  2. Return (f^{-1}, f).
- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $m' \leftarrow \mathsf{Pad}(m, \mathsf{ID}_A, \mathsf{ID}_B)$
  2. Apply f_B and f_A^{-1} to m', to get u.
  3. Return u.
- $\mathsf{VerDec}(u, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. Apply f_B^{-1} and f_A to u, to get m'.
  2. $m \leftarrow \mathsf{Depad}(m', \mathsf{ID}_A, \mathsf{ID}_B)$.
  3. Return m.

The trapdoor functions f_A and f_B are each defined over the set $\{0,1\}^l$ for some l. (Most often both f_A and f_B will use the same value of l.)

First we consider step 2 of the signcryption function, where the trapdoor permutations are applied. Since two trapdoor permutations are involved, there are several ways in which they can be applied. Dodis et al. [5] consider the following three ways, which they call *modes of operation*. Each of them considers m' as two parts, w and s. That is, $m' = (w||s)$.

- Parallel mode: $u \leftarrow f_B(w)||f_A^{-1}(s)$. This mode has the advantage that the two trapdoor permutations can be applied in parallel, but the disadvantage that the minimum ciphertext length is twice the output size of the trapdoor permutations. A further advantage is that f_A and f_B can be of different lengths.

- Sequential mode: $u \leftarrow f_B(f_A^{-1}(w||s))$. In this case, the trapdoor permutations must be applied sequentially, but the minimum ciphertext length is half that of the parallel method.

- Extended sequential mode: $u \leftarrow f_B(f_A^{-1}(w))||s$. This mode is a slight modification of the sequential mode. Its minimum ciphertext length is only slightly longer than that of the sequential method (since s can be chosen to be very short), while it admits a much tighter security proof.

Next we turn our attention to the padding step of the signcryption function. As it turns out, there is a single padding scheme which can be used with all three of the above modes (parallel, sequential, and extended sequential) by simply adjusting some parameters. It is the second of two so-called *Probabilistic Signature and Encryption Paddings* given by Dodis et al. [5], and thus called PSEP2. It is defined as follows:

- $\mathsf{Pad}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $(m_1||m_2) \leftarrow m$
  2. $r \leftarrow^R \{0,1\}^{|d|-|m_2|}$
  3. $c \leftarrow (m_1 \oplus \mathsf{K}(r)) \;||\; \mathsf{K}'(m_2||r)$
  4. $d \leftarrow (m_2||r)$
  5. $w \leftarrow d \oplus \mathsf{G}(\mathsf{ID}_A||\mathsf{ID}_B||c)$
  6. $s \leftarrow c \oplus \mathsf{H}(w)$
  7. Return (w, s).
- $\mathsf{Depad}(w, s, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. $c \leftarrow s \oplus \mathsf{H}(w)$
  2. $d \leftarrow w \oplus \mathsf{G}(\mathsf{ID}_A||\mathsf{ID}_B||c)$
  3. $(m_2||r) \leftarrow d$
  4. $(\alpha||\beta) \leftarrow c$
  5. If $\beta = \mathsf{K}'(m_2||r)$ continue, else return $\perp$.
  6. $m_1 \leftarrow \alpha \oplus \mathsf{K}(r)$
  7. Return $m \leftarrow (m_1||m_2)$.

Note that steps 2 through 4 of the padding operation (and steps 3 through 6 of the de-padding operation) constitute a commitment scheme, while steps 5 and 6 of the padding operation (and steps 1 and 2 of the de-padding operation) are simply two rounds of a Feistel Transform applied to (d, c) with round functions $\mathsf{G}(\mathsf{ID}_A||\mathsf{ID}_B||c)$ and $\mathsf{H}(w)$.

The parameters which may be adjusted are $|m_1|$, $|m_2|$, $|c|$, and $|d|$. These must be adjusted in accordance with the level of security desired, the lengths of f_A and f_B, and the mode of operation being used.

Note that PSEP2 is only able to handle messages of length up to $|m_1| + |m_2|$, which is typically a small value. In some applications, such as key transport, this may not be a problem. However, in other applications such as email, messages of arbitrary length must be handled. Dodis et al. [5] have provided a simple solution to this problem. They propose using a private key encryption scheme (E, D) to encrypt the message m under a randomly-chosen key K_{enc}, then signcrypting K_{enc} as before. One small modification is necessary; namely, to prevent the ciphertext from tampering we include it in the hash computation in step 5 of the padding operation and step 2 of the de-padding operation. This gives the following signcryption scheme:

- $\mathsf{SigEnc}(m, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. Choose a symmetric encryption key K_{enc} at random.
  2. $c' \leftarrow \mathsf{E}_{K_{enc}}(m)$
  3. $(m_1||m_2) \leftarrow K_{enc}$
  4. $r \leftarrow^R \{0,1\}^{|d|-|m_2|}$
  5. $c \leftarrow (m_1 \oplus \mathsf{K}(r)) \;||\; \mathsf{K}'(m_2||r)$
  6. $d \leftarrow (m_2||r)$
  7. $w \leftarrow d \oplus \mathsf{G}(c'||\mathsf{ID}_A||\mathsf{ID}_B||c)$
  8. $s \leftarrow c \oplus \mathsf{H}(w)$
  9. Apply f_B and f_A^{-1} to (w, s) to get u.
  10. Return (c', u).
- $\mathsf{VerDec}(c', u, \mathsf{ID}_A, \mathsf{ID}_B)$:
  1. Apply f_B^{-1} and f_A to u to get (w, s).
  2. $c \leftarrow s \oplus \mathsf{H}(w)$
  3. $d \leftarrow w \oplus \mathsf{G}(c'||\mathsf{ID}_A||\mathsf{ID}_B||c)$
  4. $(m_2||r) \leftarrow d$
  5. $(\alpha||\beta) \leftarrow c$
  6. If $\beta = \mathsf{K}'(m_2||r)$ continue, else return $\perp$.
  7. $m_1 \leftarrow \alpha \oplus \mathsf{K}(r)$

8. $K_{enc} \leftarrow (m_1||m_2)$
9. Return $m \leftarrow \mathsf{D}_{K_{enc}}(c')$.

In fact, this defines three different signcryption schemes, since the mode of operation (parallel, sequential, or extended sequential) is not specified.

5.3 Properties of PSEP2

PSEP2 has several advantages over other signcryption schemes we have seen so far. First of all, it provides the highest possible levels of security, namely IND-CCA2 and sUF-CMA in the insider security model.

Another advantage is that PSEP2 can be used as a plain encryption or signature scheme. By replacing f_A with the identity permutation, it becomes an IND-CCA2-secure encryption scheme, and by replacing f_B with the identity permutation, it becomes an sUF-CMA-secure signature scheme. Here we will call these schemes PSEP2-ENC and PSEP2-SIG, respectively.

Dodis et al. [5] did not explicitly consider non-repudiation, but it turns out that we can easily add it to PSEP2. To see this, consider the three possible trapdoor steps:

$$u \leftarrow f_B(w)||f_A^{-1}(s) \qquad \text{(Parallel mode)}$$
$$u \leftarrow f_B(f_A^{-1}(w||s) \qquad \text{(Sequential mode)}$$
$$u \leftarrow f_B(f_A^{-1}(w))||s \qquad \text{(Extended sequential mode)}$$

Note that in each case, the recipient can apply f_B^{-1} to the appropriate part of the ciphertext, thereby replacing f_B with the identity permutation and giving a valid PSEP2-SIG signature. That is, PSEP2 is a PSEP2-SIG-verifiable signcryption scheme, and thus provides non-repudiation.

Note that PSEP2 does not provide a significant computational advantage over trapdoor-based $\mathcal{S}t\mathcal{E}'$ (or $\mathcal{E}t\mathcal{S}'$) since both require two trapdoor operations. In fact, it is easy to see that we cannot hope to do better; clearly the sender's trapdoor permutation is needed for authenticity, and the recipient's for confidentiality. However, even though the total computational cost cannot be significantly reduced, the parallel mode of PSEP2 does allow the two trapdoor permutations to be applied in parallel, which could reduce the computation time by half.

One further advantage of the parallel mode over the other two is that it allows f_A and f_B of differing lengths. That is, each user can independently choose the length of his keys. When using the other two modes, all users must agree upon a common key length.

Chapter 6

Comparison

In this chapter we compare the signcryption schemes which were presented in the previous chapters.

6.1 Cost of Signcryption

One of the possible ways to compare signcryption schemes is in terms of their computational and communication overhead. However, there are a number of factors which could make such a comparison difficult to carry out, and even misleading. For example, the results of such a comparison could depend on:

- the choice of underlying encryption and signature schemes or trapdoor permutation family;
- the length of messages to be signcrypted;
- the level of security desired; and
- the number of oracle queries allowed to a potential adversary.

For this reason we will not attempt such a comparison here. It is worthwhile, however, to consider how much of an improvement signcryption can possibly provide. Since signcryption combines the functionalities of signature and encryption schemes, we would not expect it to perform better than either of them. This is indeed the case, since any signcryption scheme can be converted into an encryption scheme and a signature scheme, both of which have the same computational and communication overhead as the original signcryption scheme, and which satisfy the same security properties. These

"induced" encryption and signature schemes were first introduced by An, Dodis, and Rabin [1], and are defined as follows:

Definition 6.1. *Let $\mathcal{SC}$ be a signcryption scheme. Then the* induced encryption scheme *of $\mathcal{SC}$ is given by:*

- $\mathsf{Enc\text{-}KeyGen}(1^k)$*:*
  1. $(\mathsf{SDK}_A, \mathsf{VEK}_A) \leftarrow \mathsf{KeyGen}(1^k)$
  2. $(\mathsf{SDK}_B, \mathsf{VEK}_B) \leftarrow \mathsf{KeyGen}(1^k)$
  3. $pub \leftarrow \{\mathsf{VEK}_A, \mathsf{VEK}_B\}$
  4. *Return* $(\mathsf{EK}, \mathsf{DK}) \leftarrow (\{\mathsf{SDK}_A, pub\}, \{\mathsf{SDK}_B, pub\})$.
- $\mathsf{Enc}_{\mathsf{EK}}(m)$*:*
  1. *Return* $c \leftarrow \mathsf{SigEnc}_{\mathsf{SDK}_A, \mathsf{VEK}_B}(m)$.
- $\mathsf{Dec}_{\mathsf{DK}}(c)$*:*
  1. *Return* $m \leftarrow \mathsf{VerDec}_{\mathsf{VEK}_A, \mathsf{SDK}_B}(c)$.

Definition 6.2. *Let $\mathcal{SC}$ be a signcryption scheme. Then the* induced signature scheme *of $\mathcal{SC}$ is given by:*

- $\mathsf{Sig\text{-}KeyGen}(1^k)$*:*
  1. $(\mathsf{SDK}_A, \mathsf{VEK}_A) \leftarrow \mathsf{KeyGen}(1^k)$
  2. $(\mathsf{SDK}_B, \mathsf{VEK}_B) \leftarrow \mathsf{KeyGen}(1^k)$
  3. $pub \leftarrow \{\mathsf{VEK}_A, \mathsf{VEK}_B\}$
  4. *Return* $(\mathsf{SK}, \mathsf{VK}) \leftarrow (\{\mathsf{SDK}_A, pub\}, \{\mathsf{SDK}_B, pub\})$.
- $\mathsf{Sig}_{\mathsf{SK}}(m)$*:*
  1. *Return* $s \leftarrow \mathsf{SigEnc}_{\mathsf{SDK}_A, \mathsf{VEK}_B}(m)$.
- $\mathsf{Ver}_{\mathsf{VK}}(s)$*:*
  1. *If* $\mathsf{VerDec}_{\mathsf{VEK}_A, \mathsf{SDK}_B}(s) = \perp$ *return* fail*, else return* succeed.

While these induced schemes do give a lower bound for the cost of signcryption, they are not especially useful in practice, since their key sizes are much larger than those of the original signcryption scheme.

Of course there is no upper bound on the cost of signcryption, but it would make little sense to consider signcryption schemes with overhead significantly greater than the overhead of the traditional sign-then-encrypt approach. Since $\mathcal{SE}'$ provides a very strong level of security, as well as nonrepudiation, it makes sense to gauge other signcryption schemes against it.

The computational cost of $\mathcal{S}t\mathcal{E}'$ is simply the sum of the costs for its underlying encryption and signature schemes. Informally then, we have

$$\begin{aligned}\text{Cost(Encryption)}, \text{Cost(Signature)} &\leq \text{Cost(Signcryption)} \\ &\leq \text{Cost(Encryption)} + \text{Cost(Signature)}.\end{aligned}$$

This implies that we cannot expect any signcryption scheme to offer more than a two-fold improvement in computational cost over $\mathcal{S}t\mathcal{E}'$.

With respect to communication overhead, there is a bit more room for improvement. This is because the ciphertext is lengthened not only by the signature and encryption steps, but also by the sender's identity, which is appended before encryption. If the identity is hashed to minimize its length, then we have

$$\begin{aligned}\text{Cost(Encryption)}, \text{Cost(Signature)} &\leq \text{Cost(Signcryption)} \\ &\leq \text{Cost(Encryption)} + \text{Cost(Signature)} + |\,\mathsf{H}(\cdot)|.\end{aligned}$$

Here we can assume that $|\,\mathsf{H}(\cdot)|$ is less than the overhead introduced by the encryption and signature steps, so we certainly cannot expect more than a three-fold improvement in communication overhead over $\mathcal{S}t\mathcal{E}'$.

6.2 Comparison Charts

Now we turn our attention to those features which can be more easily compared. In Table 6.1 we look at the levels of security provided by each of the proposed signcryption schemes. In the table, the less-than symbol ($<$) indicates that a slightly weaker security level is achieved, as explained in Section 4.2. Question marks (???) indicate that the problem has not been explicitly considered (although in these cases it is known that indistinguishability is not provided), while a dash (—) indicates that no security is provided.

Here PSEP2 is the clear winner, with all three of its modes of operation providing a better level of security than even $\mathcal{E}t\mathcal{S}'$ and $\mathcal{S}t\mathcal{E}'$. PSEP2 has the further advantage that it achieves this level of security while requiring only trapdoor permutations, which are relatively easy to construct. $\mathcal{E}t\mathcal{S}'$ and $\mathcal{S}t\mathcal{E}'$ provide quite good security, but only if the underlying encryption and signature schemes do as well. Bao and Deng's scheme and SC-DSA+ are in a similar situation, since their UF-CMA security depends on the unproven assumption that DSA is UF-CMA-secure.

	Outsider Security		Insider Security	
	Conf.	Auth.	Conf.	Auth.
$\mathcal{E}t\mathcal{S}'$	IND-gCCA2	sUF-CMA	IND-gCCA2	sUF-CMA
$\mathcal{S}t\mathcal{E}'$	IND-CCA2	UF-CMA	IND-CCA2	UF-CMA
$\mathcal{E}\&\mathcal{S}$	???	sUF-CMA	???	sUF-CMA
Zheng	$<$ IND-CCA2	$<$ UF-CMA	—	$<$ UF-CMA
Bao & Deng	???	UF-CMA	—	UF-CMA
SC-DSA+	$<$ IND-CCA2	UF-CMA	—	UF-CMA
PSEP2	IND-CCA2	sUF-CMA	IND-CCA2	sUF-CMA

Table 6.1: Security Comparison

	Non-rep.	Sig. Ver.	F. Sec. / P. Rec.	Parallel
$\mathcal{S}t\mathcal{E}'$	Yes	Yes, any	Forward Secrecy	—
$\mathcal{E}t\mathcal{S}'$	—	—	Forward Secrecy	—
$\mathcal{E}\&\mathcal{S}$	Yes	Yes, any	Forward Secrecy	SC
Zheng	—	—	Past Recovery	—
Bao & Deng	Yes	—	Past Recovery	SC
SC-DSA+	Yes	Yes, DSA	Past Recovery	SC
PSEP2 (P)	Yes	Yes, PSEP2-SIG	Forward Secrecy	SC, DSC
PSEP2 (S)	Yes	Yes, PSEP2-SIG	Forward Secrecy	—
PSEP2 (X)	Yes	Yes, PSEP2-SIG	Forward Secrecy	—

Table 6.2: Feature Comparison

In Table 6.2 we look at which properties are offered by each of the signcryption schemes. The first column indicates whether a secure non-repudiation mechanism is available. The second column indicates whether the signcryption scheme is $\mathcal{S}$-verifiable, and if so, for which signature scheme. The third column indicates which of forward secrecy or past recovery is provided. The last column indicates whether the signcryption (SC) or de-signcryption (DSC) steps can be parallelized. The letters (P, S, X) beside PSEP2 indicate the mode of operation (parallel, sequential, extended sequential).

Once again, PSEP2 compares very favourably to the others, especially its parallel mode of operation. Its only disadvantage is that its non-repudiation mechanism involves a non-standard signature scheme (PSEP2-SIG). $\mathcal{S}t\mathcal{E}'$ is

also a good choice in cases where parallelization is not important. It has a very simple and $\mathcal{S}$-verifiable non-repudiation mechanism. In cases where past recovery is required, then clearly SC-DSA+ is the best choice.

Chapter 7

Conclusions

7.1 Lessons Learned

Although Zheng originally set out to achieve greater efficiency by combining encryption and signature schemes, the focus has subsequently shifted to other issues, such as security and non-repudiation. The issue of security is of key importance, since it has implications for all systems which use encryption and signatures together. There are many such systems in use today, and it would be worthwhile to reconsider them in terms of the security model for signcryption.

The security model has taught us what is perhaps the most important lesson to be learned from signcryption, namely that simply composing two cryptographic primitives (in this case, encryption and signature schemes) does not necessarily result in something which satisfies the security properties of both. It was only after defining an appropriate security model that we were able to pinpoint and correct the defects of the widely-used sign-then-encrypt paradigm.

7.2 Directions for Future Research

Ultimately, one would like to have a signcryption scheme which simultaneously

- provides IND-CCA2 and sUF-CMA security in the insider model;
- has computational and communication overhead not significantly greater than a conventional encryption or signature scheme;

- has key sizes not significantly greater than a conventional encryption or signature scheme;
- allows each user to choose his key size independently;
- provides a secure non-repudiation mechanism, preferably via a widely-used signature scheme;
- allows for significant parallelization in both the signcryption and de-signcryption steps; and
- provides efficient encrypt-only and sign-only modes.

Of all the signcryption schemes presented here, none of them are able to achieve all of these goals. PSEP2 (in its parallel mode) comes the closest, but it is suboptimal with respect to computational overhead, and provides non-repudiation only via its own special-purpose signature scheme. There is still plenty of room for improvement on the way to a truly optimal signcryption scheme.

Another area worthy of exploration is the use of the signcryption primitive in cryptographic protocols. The signcryption primitive is a powerful abstraction, since it encompasses not only confidentiality and authenticity, but also user identities (via its multi-user security definition). As mentioned in the introduction, this has allowed for the construction of a very simple authenticated key-exchange protocol. Undoubtedly there are other protocols which could be similarly simplified using signcryption.

Bibliography

[1] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In L.R. Knudsen, editor, *Proc. of Eurocrypt '02*, volume 2332 of *LNCS*, pages 83–107. Springer-Verlag, 2002. Updated version available at: http://theory.lcs.mit.edu/~yevgen/ps/signcrypt.ps.

[2] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In D. Naccache and P. Paillier, editors, *Proc. of PKC '02*, volume 2274 of *LNCS*, pages 80–98. Springer-Verlag, 2002.

[3] Feng Bao and Robert H. Deng. A signcryption scheme with signature directly verifiable by public key. In H. Imai and Y. Zheng, editors, *Proc. of PKC '98*, volume 1431 of *LNCS*, pages 55–59. Springer-Verlag, 1998.

[4] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[5] Yevgeniy Dodis, Michael J. Freedman, Stanislaw Jarecki, and Shabsi Walfish. Optimal signcryption from any trapdoor permutation, 2004. Cryptology ePrint Archive, Report 2004/020. http://eprint.iacr.org/2004/020/.

[6] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Proc. of NDSS '00*, pages 143–154, 2000.

[7] John Malone-Lee. Identity based signcryption, 2002. Cryptology ePrint Archive, Report 2002/098. http://eprint.iacr.org/2002/098/.

[8] Alfred Menezes and Nigel Smart. Security of signature schemes in a multi-user setting. In Dieter Jungnickel, Jennifer Key, and Peter Wild,

editors, *Designs, Codes and Cryptography*, volume 33, pages 261–274. Kluwer Academic Publishers, 2004.

[9] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In K. Kim, editor, *Proc. of PKC '01*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, 2001.

[10] Pascal Paillier. A trapdoor permutation equivalent to factoring. In H. Imai and Y. Zheng, editors, *Proc. of PKC '99*, volume 1560 of *LNCS*, pages 219–222. Springer-Verlag, 1999.

[11] H. Petersen and M. Michels. Cryptanalysis and improvement of signcryption schemes. In *IEE Proceedings – Computers and Digital Techniques*, volume 145, pages 149–151. 1998.

[12] Josef Pieprzyk and David Pointcheval. Parallel authentication and public-key encryption. In R. Safavi-Naini and J. Seberry, editors, *Proc. of ACISP '03*, volume 2727 of *LNCS*, pages 387–401. Springer-Verlag, 2003.

[13] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In U. Maurer, editor, *Proc. of Eurocrypt '96*, volume 1070 of *LNCS*, pages 387–398. Springer-Verlag, 1996.

[14] Jun-Bum Shin, Kwangsu Lee, and Kyungah Shim. New DSA-verifiable signcryption schemes. In P.J. Lee and C.H. Lim, editors, *Proc. of ICISC '02*, volume 2587 of *LNCS*, pages 35–47. Springer-Verlag, 2002.

[15] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In B.S. Kaliski Jr., editor, *Proc. of Crypto '97*, volume 1294 of *LNCS*, pages 165–179. Springer-Verlag, 1997.