# Chapter 2
# Security for Signcryption: The Two-User Model

**Jee Hea An and Tal Rabin**

## 2.1 Introduction

Signcryption is a cryptographic primitive designed to simultaneously provide confidentiality and integrity protection in a communication (see Chap. 1 for a more detailed description of the role of signcryption in a communication architecture). It is a public-key primitive and can be viewed as the public-key version of the symmetric-key primitive known as *authenticated encryption*; indeed, the two primitives share many similarities at a high level. Signcryption was originally proposed by Zheng [203, 204] with the intention that the primitive should satisfy "Cost(Signature & Encryption) $\ll$ Cost(Signature) $+$ Cost(Encryption)." This inequality can interpreted in a number of ways:

- A signcryption scheme should be more computationally efficient than a naive combination of public-key encryption and digital signatures.
- A signcryption scheme should produce a signcryption "ciphertext" which is shorter than a naive combination of a public-key encryption ciphertext and a digital signature.
- A signcryption scheme should provide greater security guarantees and/or greater functionality than a naive combination of public-key encryption and digital signatures.

Of course, we would ideally aim to produce a scheme which gave all three advantages; however, in the absence of such a scheme, any one of these advantages may be useful depending on the nature of the application for which signcryption is being used. A discussion of the potential uses of signcryption in practical applications is given in Chap. 12.

This chapter provides a formal definition for the security of signcryption in the *two-user setting* and analysis of the security of signcryption schemes that are constructed by generically composing signature and encryption schemes in the public-

T. Rabin(✉)
IBM T.J. Watson Research Center, Hawthorne, NY, USA
e-mail: talr@us.ibm.com

key setting. The first attempt to produce security models for signcryption was given by Steinfeld and Zheng [184]; however, this work only proposed a security model for the integrity protection property of a signcryption scheme. This chapter will be based on the more complete treatment given by the work of An et al. [10]. The problem of defining the security of signcryption in the public-key setting is more involved than the corresponding task in the symmetric setting [26, 117] due to the asymmetric nature of the former. The asymmetry of keys makes a difference in the notions of both authenticity and privacy on two major fronts which are addressed in this chapter.

The first difference for the public-key setting is that the security of the signcryption needs to be defined in the *multi-user* setting, where issues with users' identities need to be addressed. In contrast, authenticated encryption in the symmetric setting can be fully defined in a much simpler *two-user* setting. We argue that there is interest in the two-user setting in the public-key setting even though it does not provide all the security guarantees. There are quite a few subtle issues with defining the security of signcryption in the (simpler) two-user setting and thus starting in this setting highlights these delicate issues and is non-trivial.

The asymmetry of the public-key setting not only makes a difference in the multi-user and two-user settings but also makes a difference in the adversary's position depending on its knowledge of the keys. We give two definitions for security of signcryption depending on whether the adversary is an "outsider" (i.e., a third party who only knows the public information) or "insider" (i.e., a legal user of the network, either the sender or the receiver, or someone that knows the secret key of either the sender or the receiver). We call the former "outsider security" and the latter "insider security."

In this chapter, we will define security notions for both insider and outsider security in terms of both privacy (i.e., indistinguishability against chosen ciphertext attack, IND-CCA2) and authenticity (i.e., strong unforgeability against chosen message attack, sUF-CMA). We then analyze the security of the signcryption schemes that are constructed by generically composing signature and encryption schemes in the following three methods: Encrypt-and-Sign ($\mathcal{E}\&\mathcal{S}$), Encrypt-then-Sign ($\mathcal{E}t\mathcal{S}$), and Sign-then-Encrypt ($\mathcal{S}t\mathcal{E}$). As observed in [26, 117] in the symmetric setting, we show that the parallel $\mathcal{E}\&\mathcal{S}$ method does not provide even the weak IND-CPA security for privacy nor does it provide the strongest sUF-CMA security for authenticity (although it provides slightly weaker UF-CMA security) in either insider or outsider security models.

For the sequential $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ methods, we consider the following cases: security corresponding to the operation performed *last* (i.e., authenticity in the $\mathcal{E}t\mathcal{S}$ method and privacy in the $\mathcal{S}t\mathcal{E}$ method) and security corresponding to the operation performed *first* (i.e., privacy in the $\mathcal{E}t\mathcal{S}$ method and authenticity in the $\mathcal{S}t\mathcal{E}$ method). We show that the security of the last operation is *preserved* in both the insider and the outsider security models—that is, the $\mathcal{E}t\mathcal{S}$ method inherits the authenticity property of the base signature scheme and the $\mathcal{S}t\mathcal{E}$ method inherits the privacy property of the base encryption scheme. However, we show that the security of the first operation may or may not be preserved depending on the security models and the strengths of security considered. In the strong insider security model, the security of the first operation is *not* preserved against the strongest security notions of privacy and authenticity (i.e., IND-CCA2 and sUF-CMA security) although it is preserved against weaker security notions (e.g., IND-CPA, IND-gCCA2 [10], and UF-CMA security). In the weaker outsider security model, on the other hand, the security of the first operation can even be *amplified* as long as the security of the last operation is strong enough, exactly as in the symmetric setting [9, 26, 117].

## 2.2 Definition of Signcryption in the Two-User Setting

The definition of signcryption is a little bit more involved than the corresponding definition of authenticated encryption in the symmetric setting. Indeed, in the *symmetric* setting, we only have one specific pair of users who (1) share a single key; (2) trust each other; (3) "know who they are"; and (4) only care about being protected from "the rest of the world." In contrast, in the *public*-key setting, each user independently publishes its public keys, after which it can send/receive messages to/from any other user. In particular, (1) each user should have an explicit identity (associated with its public key); (2) each signcryption has to explicitly contain the (presumed) identities of the sender $S$ and the receiver $R$; (3) each user should be protected from every other user. As we have said, complete security notions for signcryption schemes should be defined in the *multi-user* setting. However, the two-user setting provides important insights into the subtleties of signcryption and so we will provide the definitions for the two-user setting as a gentle introduction to the subject. We will provide full multi-user security models in Chap. 3.

### *2.2.1 Two Security Notions in the Two-User Setting*

#### 2.2.1.1 Syntax

A signcryption scheme $\Pi$ consists of five algorithms, $\Pi = (\texttt{Setup}, \texttt{KeyGen}_S, \texttt{KeyGen}_R, \texttt{Signcrypt}, \texttt{Unsigncrypt})$:

- The (possibly randomized) setup algorithm Setup takes as input a security parameter $1^k$ and outputs any common parameters *param* required by the signcryption schemes. This may include the security parameter $1^k$, the description of a group $\mathbb{G}$ and a generator $g$ for that group, choices for hash functions or symmetric encryption schemes, etc.

  It is important to note that this algorithm does not output a secret key. In fact, it may be important that the algorithm does not leak any information about the common parameters *except* those values which are explicitly stated as being part of the output. The security of the scheme may be jeopardized if extra information about the common parameters is leaked; for example, if the common parameters include two group elements $g$ and $h$, then security may be jeopardized if the setup algorithm leaks the discrete logarithm of $h$ with the respect to $g$. Hence, all users must trust that the setup algorithm is computed correctly and securely.

- The randomized sender key generation algorithm $\texttt{KeyGen}_S$ takes as input the common parameters *param* and outputs a pair of keys $(sk_S, pk_S)$, where $sk_S$ is the sender's signing key, which is kept secret, and $pk_S$ is the sender's verification key pair $(pk_S, pk_R)$, which is made public; we write $(sk_S, pk_S) \overset{R}{\leftarrow} \texttt{KeyGen}_S(param)$.
- The randomized receiver key generation algorithm $\texttt{KeyGen}_R$ takes as input the common parameters *param* and outputs a pair of keys $(sk_R, pk_R)$, where $sk_R$ is the receiver's decryption key, which is kept secret, and $pk_R$ is the receiver's encryption key, which is made public; we write $(sk_R, pk_R) \overset{R}{\leftarrow} \texttt{KeyGen}_R(param)$. In a complete system, a user would have two key pairs: a pair $(sk_S, pk_S)$ which is used when the user is sending a message and a pair $(sk_R, pk_R)$ which is used when the user is receiving a message. It is possible for a user to have a single key pair $(sk, pk)$ which is used for both sending and receiving messages—this issue is discussed in depth in Sect. 5.4—but the simpler two-key presentation suits our purposes for now. We note that we may always set $pk = (pk_S, pk_R)$ and $sk = (sk_S, sk_R)$ and so our presentation is an example of the more general case.
- The randomized *signcryption* (sign/encrypt) algorithm `Signcrypt` takes as input the common parameters *param*, the sender's secret key $sk_S$, the receiver's public key $pk_R$, and a message $m$ from the associated message space $\mathcal{M}$. It internally flips some coins and outputs a signcryption ciphertext $C$; we will typically write $C \leftarrow \texttt{Signcrypt}(sk_S, pk_R, m)$ or $C \leftarrow \texttt{Signcrypt}(m)$ (omitting *param*, $sk_S$ and $pk_R$ for brevity).
- The deterministic *unsigncryption* (verify/decrypt) algorithm `Unsigncrypt` takes as input the common parameters *param*, the sender's public key $pk_S$, the receiver's secret key $sk_R$, and the signcryption ciphertext $C$. It outputs either $m \in \mathcal{M}$ or an error symbol $\perp$ which indicates that the message was not encrypted or signed properly. We write $m \leftarrow \texttt{Unsigncrypt}(pk_S, sk_R, C)$ or $m \leftarrow \texttt{Unsigncrypt}(C)$ (again, omitting the common parameters and keys).

We require that $\texttt{Unsigncrypt}(\texttt{Signcrypt}(m)) = m$ for any $m \in \mathcal{M}$.

#### 2.2.1.2 Security of Signcryption

Fix the sender $S$ and the receiver $R$. The security goal is to provide both authenticity and privacy of communicated data. In the symmetric setting, since the sender and the receiver share the same secret key, the only security model that makes sense is one in which the adversary is modeled as a third party or an *outsider* who does not know the shared secret key. However, in the public-key setting, the sender and the receiver do not share the same secret key but each has his/her own secret key. Due to this asymmetry of the secret keys, we need to protect the data not only from an outsider but also from an *insider* who is a legal user of the system (i.e., the sender or the receiver themselves or someone who knows either the sender's secret key or the receiver's secret key). Hence, we have an additional security notion in the public-key setting and we call it *insider security*. As opposed to the insider security, we call the security against an outsider *outsider security*, which is the security that is also considered in the symmetric setting.

## Outsider Security

We define security against the strongest security notions of authenticity (the analogs of UF-CMA or sUF-CMA for digital signature schemes) and privacy (the analog of IND-CCA2 for public-key encryption schemes). Weaker notions could easily be defined as well. We assume that the adversary $\mathcal{A}$ has the public information $(pk_S, pk_R)$. It also has oracle access to the functionalities of both the sender and the receiver. Specifically, it can mount a chosen message attack on the sender by asking the sender to produce a signcryption $C$ of an arbitrary message $m$. In other words, $\mathcal{A}$ has access to the *signcryption oracle*. Similarly, it can mount a chosen ciphertext attack on the receiver by giving the receiver any candidate signcryption $C$ and receiving back the message $m$ (where $m$ could be $\perp$), i.e., $\mathcal{A}$ has access to the *unsigncryption oracle*. Notice, $\mathcal{A}$ cannot by itself run either the signcryption or the unsigncryption oracles due to the lack of corresponding secret keys $sk_S$ and $sk_R$.

To break the UF-CMA security of the signcryption scheme, $\mathcal{A}$ has to come up with a *valid* signcryption $C$ (i.e., a ciphertext $C$ for which the unsigncryption oracle does not return $\perp$) of a "new" message $m$, which it did not ask the sender to signcrypt earlier (note that $\mathcal{A}$ is not required to "know" $m$ when producing $C$ although $\mathcal{A}$ can always compute $m$ by querying the unsigncryption oracle with $C$). The signcryption scheme is said to be *outsider secure* in the UF-CMA sense if any PPT $\mathcal{A}$ has a negligible chance of succeeding in the UF-CMA attack. To break the sUF-CMA security of the signcryption scheme, $\mathcal{A}$ has to come up with a valid signcryption $C$ which was not returned by the sender earlier (note that $C$'s unsigncryption output $m$ does not have to be "new"). Formally, we consider a game played between a hypothetical challenger and a PPT attacker $\mathcal{A}$:

1. The challenger generates common parameters $param \overset{R}{\leftarrow} \texttt{Setup}(1^k)$, a sender key pair $(sk_S, pk_S) \overset{R}{\leftarrow} \texttt{KeyGen}_S(param)$, and a receiver key pair $(sk_R, pk_R) \overset{R}{\leftarrow} \texttt{KeyGen}_R(param)$.
2. The attacker runs $\mathcal{A}$ on the input $(param, pk_S, pk_R)$. The attacker may query a signcryption oracle with a message $m \in \mathcal{M}$ to receive the signcryption ciphertext $C \overset{R}{\leftarrow} \texttt{Signcrypt}(sk_S, pk_R, m)$. The attacker may also query an unsigncryption oracle with a ciphertext $C$ to receive the message $m \leftarrow \texttt{Unsigncrypt}(pk_S, sk_R, C)$. The attacker terminates with the output of a ciphertext $C$.

The attacker wins the UF-CMA game if (1) $m \leftarrow \texttt{Unsigncrypt}(pk_S, sk_R, C)$ satisfies $m \neq \perp$ and (2) if $m$ was never submitted to the signcryption oracle. The attacker wins the sUF-CMA game if (1) $m \leftarrow \texttt{Unsigncrypt}(pk_S, sk_R, C)$ satisfies $m \neq \perp$ and (2) the signcryption oracle never returned $C$. The signcryption scheme is said to be *outsider secure* in the (s)UF-CMA sense if every PPT attacker $\mathcal{A}$ has a negligible chance of succeeding in the (s)UF-CMA attack.

To break the IND-CCA2 security of the signcryption scheme, $\mathcal{A}$ has to come up with two equal-length messages $m_0$ and $m_1$. One of these will be signcrypted at random, the corresponding signcryption challenge $C^*$ will be given to $\mathcal{A}$ and $\mathcal{A}$ has to guess which message was signcrypted. Here, $\mathcal{A}$ is forbidden to query the unsigncryption oracle on the challenge $C^*$. Formally, we consider a game played between a challenger and a PPT attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

1. The challenger generates common parameters $param \overset{R}{\leftarrow} \texttt{Setup}(1^k)$, a sender key pair $(sk_S, pk_S) \overset{R}{\leftarrow} \texttt{KeyGen}_S(param)$, and a receiver key pair $(sk_R, pk_R) \overset{R}{\leftarrow} \texttt{KeyGen}_R(param)$.
2. The attacker runs $\mathcal{A}_1$ on the input $(param, pk_S, pk_R)$. The attacker may query a signcryption oracle with a message $m \in \mathcal{M}$ to receive the signcryption ciphertext $C \overset{R}{\leftarrow} \texttt{Signcrypt}(sk_S, pk_R, m)$. The attacker may also query an unsigncryption oracle with a ciphertext $C$ to receive the message $m \leftarrow \texttt{Unsigncrypt}(pk_S, sk_R, C)$. The attacker terminates with the output of two equal-length messages $m_0, m_1 \in \mathcal{M}$ and some state information $\alpha$.
3. The challenger chooses $b \overset{R}{\leftarrow} \{0, 1\}$ and computes the challenge ciphertext $C^* \overset{R}{\leftarrow} \texttt{Signcrypt}(sk_S, pk_R, m_b)$.
4. The attacker runs $\mathcal{A}_2$ on the input of the challenge ciphertext $C^*$ and the state information $\alpha$. The attacker may query the signcryption and unsigncryption oracles as before, with the exception that the attacker is forbidden from submitting the ciphertext $C^*$ to the unsigncryption oracle. The attacker terminates with the output of a bit $b'$.

The attacker wins if $b = b'$ and the attacker's advantage is defined to be

$$\varepsilon = |Pr[b = b'] - 1/2|$$

The signcryption scheme is said to be *outsider secure* in the IND-CCA2 sense if every PPT attacker $\mathcal{A}$ has a negligible advantage in the IND-CCA2 attack.

### Insider Security

Security notions for insider security are similar to those for outsider security, except that the attacker is given one of the private keys of the users. In the (s)UF-CMA game, the attacker is given the private key of the receiver, indicating that the attacker is the receiver and that the signcryption scheme prevents a receiver from forging a signcryption ciphertext that purports to be from the sender. This is a necessary condition if non-repudiation is to be achieved. In the IND-CCA2 game, the attacker is given the private key of the sender, indicating that the attacker is the sender and that the signcryption scheme prevents a sender from deciphering a signcryption ciphertext that has previously been produced. This means that the signcryption scheme protects the confidentiality of messages even if the sender's private key is subsequently leaked to an attacker.

The formal model for insider (s)UF-CMA security is as follows:

1. The challenger generates common parameters $param \xleftarrow{R} \mathtt{Setup}(1^k)$, a sender key pair $(sk_S, pk_S) \xleftarrow{R} \mathtt{KeyGen}_S(param)$, and a receiver key pair $(sk_R, pk_R) \xleftarrow{R} \mathtt{KeyGen}_R(param)$.
2. The attacker runs $\mathcal{A}$ on the input $(param, pk_S, sk_R, pk_R)$. The attacker may query a signcryption oracle with a message $m \in \mathcal{M}$ to receive the signcryption ciphertext $C \xleftarrow{R} \mathtt{Signcrypt}(sk_S, pk_R, m)$. (The attacker need not be given access to an unsigncryption oracle as it can compute the unsigncryption algorithm itself using $sk_R$.) The attacker terminates with the output of a ciphertext $C$.

The attacker wins the UF-CMA game if (1) $m \leftarrow \mathtt{Unsigncrypt}(pk_S, sk_R, C)$ satisfies $m \neq \perp$ and (2) if $m$ was never submitted to the signcryption oracle. The attacker wins the sUF-CMA game if (1) $m \leftarrow \mathtt{Unsigncrypt}(pk_S, sk_R, C)$ satisfies $m \neq \perp$ and (2) the signcryption oracle never returned $C$. The signcryption scheme is said to be *insider secure* in the (s)UF-CMA sense if every PPT attacker $\mathcal{A}$ has a negligible chance of succeeding in the (s)UF-CMA attack.

The formal model for insider IND-CCA2 security is as follows:

1. The challenger generates common parameters $param \xleftarrow{R} \mathtt{Setup}(1^k)$, a sender key pair $(sk_S, pk_S) \xleftarrow{R} \mathtt{KeyGen}_S(param)$, and a receiver key pair $(sk_R, pk_R) \xleftarrow{R} \mathtt{KeyGen}_R(param)$.
2. The attacker runs $\mathcal{A}_1$ on the input $(param, pk_S, sk_S, pk_R)$. The attacker may query an unsigncryption oracle with a ciphertext $C$ to receive the message $m \leftarrow \mathtt{Unsigncrypt}(pk_S, sk_R, C)$. (Again, the attacker does not need to be given access to a signcryption oracle as it can compute the signcryption functionality using $sk_S$.) The attacker terminates with the output of two equal-length messages $m_0, m_1 \in \mathcal{M}$ and some state information $\alpha$.
3. The challenger chooses $b \xleftarrow{R} \{0, 1\}$ and computes the challenge ciphertext $C^* \xleftarrow{R} \mathtt{Signcrypt}(sk_S, pk_R, m_b)$.
4. The attacker runs $\mathcal{A}_2$ on the input of the challenge ciphertext $C^*$ and the state information $\alpha$. The attacker may query the unsigncryption oracle as before, with the exception that the attacker is forbidden from submitting the ciphertext $C^*$ to the unsigncryption oracle. The attacker terminates with the output of a bit $b'$.

The attacker wins if $b = b'$ and the attacker's advantage is defined to be

$$\varepsilon = |\Pr[b = b'] - 1/2|$$

The signcryption scheme is said to be *insider secure* in the IND-CCA2 sense if every PPT attacker $\mathcal{A}$ has a negligible advantage in the IND-CCA2 attack.

We also present an equivalent, but more elegant, definition of the insider security model. This more elegant treatment is rarely used in practice but does highlight the relationship between signcryption schemes and the related concepts of public-key encryption and digital signatures. We note that given any signcryption scheme $\Pi = (\texttt{Setup}, \texttt{KeyGen}_S, \texttt{KeyGen}_R, \texttt{Signcrypt}, \texttt{Unsigncrypt})$, we can define a corresponding *induced* signature scheme $\mathcal{S} = (\texttt{SigKeyGen}, \texttt{Sign}, \texttt{Verify})$ and encryption scheme $\mathcal{E} = (\texttt{EncKeyGen}, \texttt{Encrypt}, \texttt{Decrypt})$:

- *Signature scheme* $\mathcal{S}$. The key generation algorithm SigKeyGen runs $param \overset{R}{\leftarrow} \texttt{Setup}(1^k)$, $(sk_S, pk_S) \overset{R}{\leftarrow} \texttt{KeyGen}_S(param)$, and $(sk_R, pk_R) \overset{R}{\leftarrow} \texttt{KeyGen}_R(param)$. We set the signing key to $sk^{sig} = (param, sk_S, pk_S, pk_R)$ and the verification key to $pk^{sig} = (param, pk_S, sk_R, pk_R)$, namely, the public verification key (available to the adversary) *contains the secret key of the receiver* $R$. To sign a message $m$, $\texttt{Sign}(m)$ outputs $C = \texttt{Signcrypt}(m)$, while the verification algorithm $\texttt{Verify}(C)$ runs $m \leftarrow \texttt{Unsigncrypt}(C)$ and outputs $\top$ if and only if $m \neq \bot$. We note that the verification is indeed polynomial time since $pk^{sig}$ includes $sk_R$.
- *Encryption scheme* $\mathcal{E}$. The key generation algorithm EncKeyGen runs $param \overset{R}{\leftarrow} \texttt{Setup}(1^k)$, $(sk_S, pk_S) \overset{R}{\leftarrow} \texttt{KeyGen}_S(param)$, and $(sk_R, pk_R) \overset{R}{\leftarrow} \texttt{KeyGen}_R(param)$. We set the encryption key to $pk^{enc} = (param, sk_S, pk_S, pk_R)$ and the decryption key to $sk^{enc} = (param, pk_S, sk_R, pk_R)$, namely the public encryption key (available to the adversary) *contains the secret key of the sender* $S$. To encrypt a message $m$, $\texttt{Encrypt}(m)$ outputs $C = \texttt{Signcrypt}(m)$, while the decryption algorithm $\texttt{Decrypt}(C)$ simply outputs $\texttt{Unsigncrypt}(C)$. We note that the encryption is indeed polynomial time since $pk^{enc}$ includes $sk_S$.

The signcryption scheme is insider (s)UF-CMA secure if the induced signature scheme is (s)UF-CMA secure. The signcryption scheme is insider IND-CCA2 secure if the induced encryption scheme is IND-CCA2 secure.

### *2.2.2 Discussions on the Security Notions*

#### 2.2.2.1 Should we Require Non-Repudiation?

We note that the conventional notion of digital signatures supports *non-repudiation*. Namely, the receiver $R$ of a correctly generated signature $s$ of the message $m$ can hold the sender $S$ responsible for the contents of $m$. Indeed, presenting $s$ to a third party is sufficient for $R$ to prove that $m$ was indeed signed by $S$ as long as the signature scheme that is used to generate $s$ is unforgeable and publicly verifiable. On the other hand, non-repudiation does not *automatically* follow from the definition of signcryption. Although signcryption allows the *receiver* to be convinced that $m$ was sent by $S$, it does not necessarily enable a third party to verify this fact because the verification of the authenticity of the message $m$ may involve the receiver's secret key, depending on how the signcryption scheme is built.

We believe that non-repudiation should not be part of the *definition* of signcryption security because the necessity of this property varies depending on the applications. Indeed, non-repudiation might be needed in some applications, while explicitly undesirable in others (e.g., this issue is the essence of undeniable [58] and chameleon [119] signature schemes). We will therefore not discuss this issue any further in this chapter. The issue of non-repudiation in signcryption schemes is discussed further in Sects. 4.6 and 6.5.

#### 2.2.2.2 Insider vs. Outsider Security

We illustrate some of the differences between insider and outsider security. For example, insider security for authenticity implies non-repudiation "in principle." Namely, non-repudiation is certain at least when the receiver $R$ is willing to reveal its secret key $sk_R$ (since this induces a regular signature scheme) and may be possible by other means (e.g., with the use of an appropriate zero-knowledge proof). In contrast, outsider security leaves open the possibility that the receiver $R$ can generate—using its secret key—valid signcryptions of messages that were not actually sent by the sender $S$. In such a case, non-repudiation cannot be achieved no matter what the receiver $R$ does.

Despite the above issues, however, it might still seem that the distinction between insider and outsider security is a bit contrived, especially for privacy. Intuitively, outsider security protects the privacy of the receiver $R$ from outside intruders who do not know the secret key of the sender $S$. On the other hand, insider security assumes that the sender $S$ *is* the intruder attacking the privacy of the receiver $R$. But since the sender $S$ is the *only* user that can send valid signcryptions from $S$ to $R$, this seems to make little sense. Similarly for authenticity, if non-repudiation is *not* an issue, then insider security seems to make little sense as it assumes that the receiver $R$ is the intruder attacking the authenticity of the sender $S$, and, simultaneously, the *only* user that needs to be convinced of the authenticity of the (received) data. In many settings *outsider security might be all one needs* for privacy and/or authenticity. Still, there are some cases where the extra strength of the insider security might be important. For example, assume an adversary $\mathcal{A}$ happens to steal the key of the sender $S$. Even though now $\mathcal{A}$ can send forged messages "from $S$ to $R$," we still might not want $\mathcal{A}$ to understand previous (or even future) recorded signcryptions sent from the honest sender $S$ to the receiver $R$. Similarly, if an adversary $\mathcal{A}$ happens to steal the key of the receiver $R$, we still might not want $\mathcal{A}$ to send forged messages "from $S$ to $R$," although $\mathcal{A}$ can now understand signcryption messages sent from the honest sender $S$ to the receiver $R$. Insider security will meet these security requirements, while the outsider security might not.

Finally, we note that *achieving* outsider security could be significantly easier than insider security. One such example will be seen in Theorems 2.3 and 2.4. Another example is given by An [7] and shows that authenticated encryption in the *symmetric setting* could be used to build outsider secure signcryption, but not insider secure signcryption. A final example is the outsider secure signcryption KEM produced by Dent [73] which is discussed in Sect. 7.3. In summary, one should carefully examine if one really needs the extra guarantees of insider security.

## 2.3 Generic Compositions of Signature and Encryption

In this section, we discuss three methods of constructing signcryption schemes that are based on generic composition of signature and encryption: Encrypt-and-Sign ($\mathcal{E}\&\mathcal{S}$), Sign-then-Encrypt ($\mathcal{S}t\mathcal{E}$), and Encrypt-then-Sign ($\mathcal{E}t\mathcal{S}$).

### 2.3.1 Construction

Let $\mathcal{E} = (\texttt{EncKeyGen}, \texttt{Encrypt}, \texttt{Decrypt})$ be an encryption scheme and $\mathcal{S} = (\texttt{SigKeyGen}, \texttt{Sign}, \texttt{Verify})$ be a signature scheme. All three methods use the same common parameter algorithm and key generation algorithms—see Fig. 2.1. Essentially, the schemes require no common parameters, while the sender and receiver key generation algorithms are the key generation algorithms for the signature and encryption schemes, respectively. The three construction methods are the "Encrypt-and-Sign" ($\mathcal{E}\&\mathcal{S}$) method—see Fig. 2.2; the "Encrypt-then-Sign" ($\mathcal{E}t\mathcal{S}$) method—see Fig. 2.3; and the "Sign-then-Encrypt" ($\mathcal{S}t\mathcal{E}$) method—see Fig. 2.4.

```
$\texttt{Setup}(1^k)$:
  $param \leftarrow 1^k$
  Return $param$

$\texttt{KeyGen}_S(param)$:
  $(sk^{sig}, pk^{sig}) \xleftarrow{R} \texttt{SigKeyGen}(1^k)$
  $(sk_S, pk_S) \leftarrow (sk^{sig}, pk^{sig})$
  Return $(sk_S, pk_S)$

$\texttt{KeyGen}_R(param)$:
  $(sk^{enc}, pk^{enc}) \xleftarrow{R} \texttt{EncKeyGen}(1^k)$
  $(sk_R, pk_R) \leftarrow (sk^{enc}, pk^{enc})$
  Return $(sk_R, pk_R)$
```

**Fig. 2.1** The key generation algorithms for the generic compositions

```
$\texttt{Signcrypt}(sk_S, pk_R, m)$:
  Parse $sk_S$ as $sk^{sig}$
  Parse $pk_R$ as $pk^{enc}$
  $c \xleftarrow{R} \texttt{Encrypt}(pk^{enc}, m)$
  $\sigma \xleftarrow{R} \texttt{Sign}(sk^{sig}, m)$
  $C \leftarrow (c, \sigma)$
  Return $C$

$\texttt{Unsigncrypt}(pk_S, sk_R, C)$:
  Parse $pk_S$ as $pk^{sig}$
  Parse $sk_R$ as $sk^{enc}$
  Parse $C$ as $(c, \sigma)$
  $m \leftarrow \texttt{Decrypt}(sk^{enc}, c)$
  If $\texttt{Verify}(pk^{sig}, m, \sigma) = \perp$ then return $\perp$
  Otherwise return $m$
```

**Fig. 2.2** The Encrypt-and-Sign ($\mathcal{E}\&\mathcal{S}$) scheme

```
$\texttt{Signcrypt}(sk_S, pk_R, m)$:
  Parse $sk_S$ as $sk^{sig}$
  Parse $pk_R$ as $pk^{enc}$
  $c \xleftarrow{R} \texttt{Encrypt}(pk^{enc}, m)$
  $\sigma \xleftarrow{R} \texttt{Sign}(sk^{sig}, c)$
  $C \leftarrow (c, \sigma)$
  Return $C$

$\texttt{Unsigncrypt}(pk_S, sk_R, C)$:
  Parse $pk_S$ as $pk^{sig}$
  Parse $sk_R$ as $sk^{enc}$
  Parse $C$ as $(c, \sigma)$
  If $\texttt{Verify}(pk^{sig}, m, \sigma) = \perp$ then return $\perp$
  $m \leftarrow \texttt{Decrypt}(sk^{enc}, c)$
  Return $m$
```

**Fig. 2.3** The Encrypt-then-Sign ($\mathcal{E}t\mathcal{S}$) scheme

```
Signcrypt($sk_S$,$pk_R$,$m$):
  Parse $sk_S$ as $sk^{sig}$
  Parse $pk_R$ as $pk^{enc}$
  $\sigma \overset{R}{\leftarrow}$ Sign($sk^{sig}$,$m$)
  $C \overset{R}{\leftarrow}$ Encrypt($pk^{enc}$,$m\|\sigma$)
  Return $C$

Unsigncrypt($pk_S$,$sk_R$,$C$):
  Parse $pk_S$ as $pk^{sig}$
  Parse $sk_R$ as $sk^{enc}$
  $m\|\sigma \leftarrow$ Decrypt($sk^{enc}$,$C$)
  If Verify($pk^{sig}$,$m$,$\sigma$) $= \perp$ then return $\perp$
  Otherwise return $m$
```

**Fig. 2.4** The Sign-then-Encrypt ($\mathcal{S}t\mathcal{E}$) scheme

### *2.3.2 Security of the Parallel Composition Method*

Among the above three generic composition methods, the "Encrypt-and-Sign" ($\mathcal{E}\&\mathcal{S}$) method allows computing encryption and signature in parallel, while in the other two methods, they are computed sequentially. However, in terms of security, it is easy to see that $\mathcal{E}\&\mathcal{S}$ does *not* preserve privacy since the signature will reveal information about the message $m$ (regardless of whether the adversary is an insider or an outsider). To be more formal, we give an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the IND-CCA2 property of the signcryption scheme. The attack works in two phases:

- The attacker $\mathcal{A}_1$ outputs two distinct equal-length messages from the message space $(m_0, m_1)$.

The challenger randomly signcrypts one message to give a challenge ciphertext $(c^*, \sigma^*) \overset{R}{\leftarrow}$ Signcrypt($sk_S, pk_R, m_b$). This challenge ciphertext is given to the attacker.

- The attacker $\mathcal{A}_2$ checks whether $\sigma^*$ is a valid signature on $m_0$ or $m_1$ by computing Sign($pk_S, m_0, \sigma^*$) and Sign($pk_S, m_1, \sigma^*$). The attacker returns the appropriate bit $b$.

This may seem like a technicality, but the prospect of the digital signature leaking information about the message is very real. There is no requirement on the digital signature to preserve the confidentiality of the message. Indeed, digital signatures with message recovery, discussed in Sect. 1.3.2, guarantee that the signature will reveal the underlying message. These signature schemes still meet the strong notions of (s)UF-CMA security, but have absolutely no confidentiality properties.

Although $\mathcal{E}\&\mathcal{S}$ does not preserve privacy, it is easy to see that it preserves the UF-CMA security. Intuitively, if an adversary against the UF-CMA security of the signcryption scheme built using $\mathcal{E}\&\mathcal{S}$ succeeds, it means it succeeded in forging a signature for a "new" message, which is exactly what it means to break the UF-CMA security of the underlying signature scheme. However, for the sUF-CMA security (a stronger authenticity property), the $\mathcal{E}\&\mathcal{S}$ method does not necessarily yield a secure signcryption scheme for a similar reason as in the privacy case (both the encryption part and the signature part need to be unforgeable). Notice that these results hold in both insider and outsider security models.

### *2.3.3 Security of the Sequential Composition Methods*

In the strong insider security model, where the adversary knows all of the secret keys except for the one being attacked, signcryption security can only be based on the security of the underlying component whose secret key is unknown to the adversary. For example, in the case of confidentiality, the only key that the adversary does not know is the private key of the encryption scheme. In other words, the privacy of the signcryption scheme can only be based on the security of the public-key encryption scheme. Similarly, the integrity protection property of the signcryption scheme can only be based on the security of the digital signature scheme. Hence, preserving the security property of the underlying component is the best we can hope to achieve with insider security. However, we show that this may not always be achieved—that is, in the sequential composition methods (i.e., $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$), depending on the order of composition and the strength of the security property considered, the security of the underlying component may or may not be preserved. We show this difference by dividing the security into two cases depending on whether we consider the signcryption security property corresponding to the operation performed *first* or *last*.

When we consider the security of signcryption corresponding to the security of the operation performed *last* (i.e., authenticity in the $\mathcal{E}t\mathcal{S}$ method and privacy in the $\mathcal{S}t\mathcal{E}$ method), the security of the base component is preserved. In other words, the security of the last operation is inherited by the signcryption scheme—that is, the $\mathcal{E}t\mathcal{S}$ method inherits the authenticity of the base signature scheme and the $\mathcal{S}t\mathcal{E}$ method inherits the privacy of the base encryption scheme. Notice that in this case the security of the signcryption scheme does *not* depend on the security of the other component (i.e., the operation performed first). This is true regardless of the security models (i.e., regardless of whether we consider the insider or outsider security model).

If we consider the signcryption security corresponding to the security of the operation performed *first* (i.e., privacy in the $\mathcal{E}t\mathcal{S}$ method and authenticity in the $\mathcal{S}t\mathcal{E}$ method), then results differ depending on the security models and the composition methods. In the insider security model, the security of the first operation is *not* preserved against the strongest security notions of privacy and authenticity (i.e., IND-CCA2 security and sUF-CMA security) although it is preserved against weaker security notions (e.g., IND-CPA, IND-gCCA2 [10], and UF-CMA security). This is because the adversary who knows the secret key of the other component (i.e., the signature scheme in the $\mathcal{E}t\mathcal{S}$ method and the encryption scheme in the $\mathcal{S}t\mathcal{E}$ method) can manipulate the given signcryption ciphertext by re-signing it and submitting the modified ciphertext as a unsigncryption oracle query (in the attack against the IND-CCA2 security of the $\mathcal{E}t\mathcal{S}$ method) or re-encrypting it and submit the modified ciphertext as a forgery (in the attack against the sUF-CMA security of the $\mathcal{S}t\mathcal{E}$ method). Intuitively, this tells us that achieving the strongest security corresponding to the security of the operation performed first is not possible when the adversary knows the secret key of the operation performed last.

However, in the outsider security model (where the adversary does not know any secret keys) the results are quite different. The security of the operation performed last can help enhance the security of the operation performed first—that is, a security property stronger than that of the first operation can be achieved as long as the security of the last operation is strong enough. Indeed, it turns out that, for the $\mathcal{E}t\mathcal{S}$ method, IND-CCA2 security can be achieved from the IND-CPA security of the base encryption scheme (which is the first operation) with the help of the sUF-CMA security of the base signature scheme (which is the last operation). For the $\mathcal{S}t\mathcal{E}$ method, sUF-CMA security can be achieved from the UF-NMA security of the base signature scheme (which is the first operation) with the help of the IND-CCA2 security of the base encryption scheme (which is the last operation).

We now summarize the results in the following theorems. Theorem 2.1 states that the signcryption security corresponding to the security of the last operation is preserved in both insider and outsider security models. In order to show that, we consider only the strongest security notions (i.e., insider IND-CCA2 security for privacy and insider sUF-CMA security for authenticity) as representative cases since the proofs for other weaker notions are very similar except a few minor definitional differences.

**Theorem 2.1** *If $\mathcal{S}$ is sUF-CMA secure, then the signcryption scheme $\Pi$ built using the $\mathcal{E}t\mathcal{S}$ method is sUF-CMA secure in the insider security model. If $\mathcal{E}$ is IND-CCA2 secure, then the signcryption scheme $\Pi$ built using the $\mathcal{S}t\mathcal{E}$ method is IND-CCA2 secure in the insider security model.*

*Proof* (1) sUF-CMA security of $\mathcal{E}t\mathcal{S}$ in the insider security model

Let $\mathcal{A}'$ be a forger against the sUF-CMA security of $\Pi$ built using the $\mathcal{E}t\mathcal{S}$ method in the insider security model. We can easily construct a forger $\mathcal{A}$ against the sUF-CMA security of the signature scheme $\mathcal{S}$ that has identical probability of forging signatures. Let $(sk_S^{sig}, pk_S^{sig})$ be the keys of $\mathcal{S}$. Given the signing oracle `Sign` and the public verification key $pk_S^{sig}$, $\mathcal{A}$ picks a pair of encryption keys $(sk_R^{enc}, pk_R^{enc}) \overset{R}{\leftarrow} \texttt{EncKeyGen}(1^k)$. $\mathcal{A}$ then hands $(pk_S^{sig}, sk_R^{enc}, pk_R^{enc})$ to $\mathcal{A}'$ as the public key of the induced signature scheme. $\mathcal{A}$ can easily simulate the signcryption query of $\mathcal{A}'$ for any message $m'$ by first creating $e' \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R^{enc}, m')$ and then asking the signing oracle for $\mathcal{S}$ to sign $e'$. Finally, when $\mathcal{A}'$ produces a forgery $C$ for $\mathcal{E}t\mathcal{S}$, $\mathcal{A}$ outputs $C$ as well. For sUF-CMA security, it is easy to see that if $C$ is a valid and "new" signcryption (i.e., either the encryption part or the signature part is new), then $C$ is a valid and "new" signature too (i.e., either the message part or the signature part is new).

(2) IND-CCA2 security of $\mathcal{S}t\mathcal{E}$ in the insider security model

Let $\mathcal{A}'$ be a distinguisher against the IND-CCA2 security of a scheme $\Pi$ built using the $\mathcal{S}t\mathcal{E}$ method in the insider security model. We can easily construct a distinguisher $\mathcal{A}$ against the IND-CCA2 security of the encryption scheme $\mathcal{E}$ as follows. Let $(sk_R^{enc}, pk_R^{enc})$ be the key pair of $\mathcal{E}$. Given the public encryption key $pk_R^{enc}$ and the decryption oracle `Decrypt`, $\mathcal{A}$ picks a pair of signing keys $(sk_S^{sig}, pk_S^{sig}) \overset{R}{\leftarrow}$ $\texttt{SigKeyGen}(1^k)$. $\mathcal{A}$ then hands $(sk_S^{sig}, pk_S^{sig}, pk_R^{enc})$ to $\mathcal{A}'$, as the public key of the induced encryption scheme. To simulate the unsigncryption query $C'$ made by $\mathcal{A}'$,

$\mathcal{A}$ first decrypts $C'$ into $m'\|\sigma'$ using its own decryption oracle and then checks if $\sigma'$ is a valid signature of $m'$ and returns $m'$ if $s'$ is valid and $\perp$ if not. Next, when $\mathcal{A}'$ outputs a pair of messages $m_0$ and $m_1$, $\mathcal{A}$ outputs $m_0\|s_0$ and $m_1\|s_1$, where $s_i = \mathtt{Sign}(sk_S^{sig}, m_i)$. $\mathcal{A}$ gives $\mathcal{A}'$ the same challenge $C = \mathtt{Encrypt}(pk_R^{enc}, m_b\|s_b)$ it gets. Finally, $\mathcal{A}$ outputs the same guess $b'$ that $\mathcal{A}'$ outputs. It is easy to see that $\mathcal{A}$ has the same probability of being correct as $\mathcal{A}'$ has. □

The following theorem states that when considering the signcryption security corresponding to the operation performed first, the strongest security properties (IND-CCA2 security for $\mathcal{E}t\mathcal{S}$ and sUF-CMA security for $\mathcal{S}t\mathcal{E}$) cannot be achieved in the insider security model.

**Theorem 2.2** *Let $\mathcal{E}$ be any encryption scheme and $\mathcal{S}$ be a probabilistic signature scheme, then the signcryption scheme $\Pi$ built using the $\mathcal{E}t\mathcal{S}$ method is not IND-CCA2 secure in the insider security model. Let $\mathcal{S}$ be any signature scheme and $\mathcal{E}$ be a probabilistic encryption scheme, then the signcryption scheme $\Pi$ built using the $\mathcal{S}t\mathcal{E}$ method is not sUF-CMA secure in the insider security model.*

*Proof* (1) $\mathcal{E}t\mathcal{S}$ is *not* IND-CCA2 secure in insider security model

We show that the $\mathcal{E}t\mathcal{S}$ method cannot achieve IND-CCA2 security in the insider security model by constructing a distinguisher $\mathcal{A}$ against the IND-CCA2 security of $\Pi$ built using the $\mathcal{E}t\mathcal{S}$ method. Let $\mathcal{S}$ and $\mathcal{E}$ be the base signature and encryption schemes whose key pairs are $(sk_S^{sig}, pk_S^{sig})$ and $(sk_R^{enc}, pk_R^{enc})$, respectively. Let $pk^{enc} = (sk_S^{sig}, pk_S^{sig}, pk_R^{enc})$ be the induced encryption key and let $sk^{enc} = (sk_R^{enc}, pk_S^{sig}, pk_R^{enc})$ be the induced decryption key. Given the induced decryption oracle $\mathtt{Decrypt}$ and the induced encryption key $pk^{enc}$, $\mathcal{A}$ picks two messages $(m_0, m_1)$, where $m_0 = 0$ and $m_1 = 1$, and then outputs them to get the challenge ciphertext $C = (c, \sigma)$. Next, $\mathcal{A}$ gets the message part $c$ and re-signs $c$ by computing a "new" signature $\sigma' \overset{R}{\leftarrow} \mathtt{Sign}(sk_S^{sig}, c)$ of $c$, where $\sigma' \neq \sigma$, and then queries the induced decryption oracle with $C' = (c, \sigma')$. Notice that since we assumed $\mathcal{S}$ is probabilistic (not deterministic), with a non-negligible probability one can find a different signature for the same message in polynomial time. Since $C' \neq C$, and $\sigma'$ is a valid signature of $c$, $\mathcal{A}$ can obtain the decryption of $c$. Once the decrypted message $m$ is obtained, $\mathcal{A}$ compares it with its own message pair $(m_0, m_1)$ and outputs the bit $b$ where $m_b = m$.

(2) $\mathcal{S}t\mathcal{E}$ is *not* sUF-CMA secure in insider security model

We show that the $\mathcal{S}t\mathcal{E}$ method cannot achieve sUF-CMA security in the insider security model by constructing a forger $\mathcal{A}$ against the sUF-CMA security of a scheme $\Pi$ built using the $\mathcal{S}t\mathcal{E}$ method. Let $\mathcal{S}$ and $\mathcal{E}$ be the base signature and encryption schemes whose key pairs are $(sk_S^{sig}, pk_S^{sig})$ and $(sk_R^{enc}, pk_R^{enc})$, respectively. Let $sk^{sig} = (sk_S^{sig}, pk_S^{sig}, pk_R^{enc})$ be the induced signature key and let $pk^{sig} = (sk_R^{enc}, pk_S^{sig}, pk_R^{enc})$ be the induced verification key. Given the induced signing oracle $\mathtt{Sign}$ and the induced verification key $pk^{sig}$, the forger $\mathcal{A}$ picks a message $m$ and queries $\mathtt{Sign}$ with $m$ to get the answer $C$. $\mathcal{A}$ then decrypts $C$ using the decryption key $sk_R^{enc}$ to get $m\|s = \mathtt{Decrypt}(sk_R^{enc}, C)$, re-encrypt $m\|s$ to get

$C' = \mathtt{Encrypt}(pk_R^{enc}, m\|s)$, where $C' \neq C$, and returns $C'$ as a forgery. Notice that since $\mathcal{E}$ is a probabilistic encryption scheme (as opposed to deterministic), with a non-negligible probability, $\mathcal{A}$ can get $C'$ in polynomial time such that $C' \neq C$ when re-encrypting $m\|s$. Since $C'$ was never returned by the signing oracle $\mathtt{Sign}$ (i.e., $C' \neq C$) and $s$ is a valid signature of $m$, $C'$ is considered as a valid forgery against sUF-CMA of $\Pi$. □

Notice the negative results in Theorem 2.2 hold regardless of the strength of the security of the base encryption and signature schemes. Intuitively, this means that the security of the first operation is not protected by the last operation because both the security goals to achieve (i.e., IND-CCA2 and sUF-CMA) and the capabilities given to the adversary (i.e., having the secret key of one of the parties in the insider security model) are very strong. Notice that if we weaken the security goals (e.g., IND-gCCA2 security [10] and UF-CMA security), then the security may be preserved, as shown in [10]. Notice also that if we weaken the capabilities given to the adversary (e.g., if it is not given the secret keys as in the outsider security model), then the security may even be amplified, as shown in the next two theorems.

Unlike the insider security model, we show that in the weaker outsider security model, it is possible to amplify the security of encryption using signatures as well as the security of signatures using encryption, exactly like in the symmetric setting [9, 26, 117]. In particular, we can obtain a IND-CCA2 secure signcryption scheme via the $\mathcal{E}t\mathcal{S}$ method from a IND-CPA secure base encryption scheme with the aid of a "strong" base signature scheme. Similarly, we can obtain the sUF-CMA security via the $\mathcal{S}t\mathcal{E}$ method from a UF-NMA secure base signature scheme with the aid of a "strong" base encryption scheme. This shows that the outsider security model in the two-user setting is quite similar to the symmetric setting: namely, from the adversarial point of view the sender and the receiver "share" the secret key $(sk_S, sk_R)$. We state this in the next two theorems. Specifically, the first theorem states that the $\mathcal{E}t\mathcal{S}$ method amplifies privacy and the second theorem states that the $\mathcal{S}t\mathcal{E}$ method amplifies authenticity.

**Theorem 2.3** *If $\mathcal{E}$ is IND-CPA secure and $\mathcal{S}$ is sUF-CMA secure, then the signcryption scheme $\Pi$ built using $\mathcal{E}t\mathcal{S}$ is IND-CCA2 secure in the outsider security model.*

*Proof* Let $\mathcal{A}'$ be the adversary breaking IND-CCA2 security of the Encrypt-then-Sign signcryption scheme $\mathcal{E}t\mathcal{S}$ in the outsider security model. Recall, $\mathcal{A}'$ only knows $(pk_S^{sig}, pk_R^{enc})$ and has access to the signcryption and the unsigncryption oracles $\mathtt{Signcrypt}$ and $\mathtt{Unsigncrypt}$. By assumption, $|\Pr[b' = b] - 1/2|$ is negligible, where the probability is taken over all the randomness needed to perform the run of $\mathcal{A}'$ (as described in Sect. 2.2), $b$ is the real index of the message being signcrypted, and $b'$ is the guess of $\mathcal{A}'$.

We define the event FORGED to be the event where the adversary $\mathcal{A}'$ manages to generate a value $C' = (c', \sigma')$ on which it calls its unsigncryption oracle $\mathtt{Unsigncrypt}$ where $C'$ satisfies the following properties:

1. $C'$ passes the signature validation step, i.e., $\mathtt{Verify}(pk_S^{sig}, c', \sigma') = \top$ and
2. $C'$ was not given to $\mathcal{A}'$ by the signcryption oracle $\mathtt{Signcrypt}$.

We split the executions of $\mathcal{A}'$ into two groups: (a) the runs in which $\mathcal{A}'$ has an event FORGED and (b) runs when no such event happens. The distinction between these two cases is that in (a) the adversary uses the unsigncryption oracle in a meaningful way. In case (b) the unsigncryption oracle can be completely simulated, i.e., either the unsigncryption oracle responds with failure or it is a query which has been asked to the signcryption oracle previously. Formally, we can show the following via an application of Bayes theorem:

$$\begin{aligned}
&|\Pr[b' = b] - 1/2|\\
&\quad = |\Pr[\mathcal{A}' \text{ WINS}] - 1/2|\\
&\quad = |\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}]\Pr[\neg\text{FORGED}]\\
&\qquad + \Pr[\mathcal{A}' \text{ WINS} \mid \text{FORGED}]\Pr[\text{FORGED}] - 1/2|\\
&\quad = |\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}](1 - \Pr[\text{FORGED}])\\
&\qquad + \Pr[\mathcal{A}' \text{ WINS} \mid \text{FORGED}]\Pr[\text{FORGED}] - 1/2|\\
&\quad = |\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}] - 1/2\\
&\qquad -(\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}] - \Pr[\mathcal{A}' \text{ WINS} \mid \text{FORGED}])\Pr[\text{FORGED}]|\\
&\quad \le |\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}] - 1/2|\\
&\qquad +|\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}] - \Pr[\mathcal{A}' \text{ WINS} \mid \text{FORGED}]|\Pr[\text{FORGED}]\\
&\quad \le |\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}] - 1/2| + \Pr[\text{FORGED}]
\end{aligned}$$

Hence, it is sufficient to bound $|\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}] - 1/2|$ and $\Pr[\text{FORGED}]$ by negligible functions to show our results. We will show these results in two cases.

Case 1: Pr[FORGED] is negligible

We show that we can construct a forger $\mathcal{A}$ which breaks sUF-CMA security of signature scheme $\mathcal{S}$ with probability at least Pr[FORGED]. The assumption that the signature scheme is sUF-CMA secure shows that Pr[FORGED] is negligible. Given the signing oracle `Sign` and the public verification key $pk_S^{sig}$, the forger $\mathcal{A}$ picks a pair of encryption keys $(sk_R^{enc}, pk_R^{enc}) \overset{R}{\leftarrow}$ $\texttt{EncKeyGen}(1^k)$. $\mathcal{A}$ then picks a random bit $b$ for the index of the message being signcrypted and hands $(pk_S^{sig}, pk_R^{enc})$ to $\mathcal{A}'$ as the public key of the signcryption scheme. For each signcryption query $m$ of $\mathcal{A}'$, $\mathcal{A}$ simulates the signcryption oracle `Signcrypt` by first encrypting $m$ using the generated encryption key $pk_R^{enc}$ to get $c' \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R^{enc}, m)$ and asking its own signing oracle `Sign` to sign $c'$ to obtain $\sigma'$. For each unsigncryption query $C' = (c', \sigma')$, $\mathcal{A}$ simulates the unsigncryption oracle by first checking $\texttt{Verify}(pk_S^{sig}, c', \sigma') = \top$ and then decrypting $c'$ using the decryption key $sk_R^{enc}$. If $C'$ is a valid signcryption ciphertext, but $C'$ was not returned by the signcryption oracle, then FORGED has occurred and $\mathcal{A}$ (correctly) outputs $(c', \sigma')$ as an sUF-CMA forgery. If $\mathcal{A}'_1$ outputs $(m_0, m_1)$ to be signcrypted in the first stage, then $\mathcal{A}$ computes the challenge ciphertext $C^*$ of the message $m_b$ using the signcryption method above. If $\mathcal{A}'$ terminates without the event FORGED occurring, then $\mathcal{A}$ terminates without output. Hence, $\mathcal{A}$ wins the sUF-CMA game if and only if FORGED occurs and so Pr[FORGED] is bounded by the success probability of $\mathcal{A}$ in winning the sUF-CMA security game against the signature scheme $\mathcal{S}$.

Case 2: $|\Pr[\mathcal{A}' \text{ WINS} \mid \neg\text{FORGED}] - 1/2|$ is negligible

First, we note that since FORGED does not occur, we have that any query $C' = (c', \sigma')$ to the unsigncryption oracle must have one of the following two forms: (a) $\texttt{Verify}_S(c') = \perp$ or (b) $C'$ was already returned by $\texttt{Signcrypt}$ on some query $m'$. For a type (a) query, the oracle's correct response is $\perp$. For a type (b) query, the oracle's correct response is $m'$. In both cases, $\mathcal{A}$ will "know" the correct response returned by the unsigncryption oracle. Overall, the unsigncryption oracle is useless: $\mathcal{A}'$ can compute all the answers by itself; hence, CPA security suffices for the encryption scheme.

Formally, we show that we can construct an adversary $\mathcal{A}$ which would break the IND-CPA security of $\mathcal{E}$. Given the public encryption key $pk_R^{enc}$, $\mathcal{A}$ picks a signature key pair $(sk_S^{sig}, pk_S^{sig})$ and gives $(pk_S^{sig}, pk_R^{enc})$ to $\mathcal{A}'$ as the public keys of the signcryption scheme. For each signcryption oracle query $m$, $\mathcal{A}$ simulates the signcryption oracle by first encrypting $m$ using the given encryption key $pk_R^{enc}$ to get $c' \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R^{enc}, m)$ and then signing $c'$ using the picked signing key $sk_S^{sig}$ to get $\sigma' \overset{R}{\leftarrow} \texttt{Sign}(sk_S^{sig}, c')$. $\mathcal{A}$ keeps track of all the tuples $(m, c', \sigma')$ that were simulated by the signcryption oracle in a table. For each unsigncryption query $C' = (c', \sigma')$, $\mathcal{A}$ returns $\perp$ to $\mathcal{A}'$ if $C'$ is a type (a) query or it returns the corresponding $m$ by using the table kept in the signcryption oracle simulation if $C'$ is a type (b) query. If $\mathcal{A}'$ outputs $(m_0, m_1)$, then $\mathcal{A}$ outputs $(m_0, m_1)$ and gets the challenge ciphertext $c$. $\mathcal{A}$ then signs $c$ to get $\sigma \overset{R}{\leftarrow} \texttt{Sign}(sk_S^{sig}, c)$ and gives $C = (c, \sigma)$ to $\mathcal{A}'$ as the challenge ciphertext. When $\mathcal{A}'$ outputs a guess bit $b'$, $\mathcal{A}$ outputs the same bit. It is clear that if FORGED does not happen, $\mathcal{A}$ simulates the correct environment for $\mathcal{A}'$. Hence $\mathcal{A}$ succeeds in the IND-CPA game against the public-key encryption scheme with overall advantage equal to that of $\mathcal{A}'$ in the IND-CCA2 game against the signcryption scheme. □

**Theorem 2.4** *If $\mathcal{E}$ is IND-CCA2 secure and $\mathcal{S}$ is UF-NMA secure, then the signcryption scheme $\Pi$ built using the $\mathcal{S}t\mathcal{E}$ method is sUF-CMA secure in the outsider security model.*

*Proof* Let $\mathcal{A}'$ be an adversary attacking sUF-CMA security of the signcryption scheme built using the $\mathcal{S}t\mathcal{E}$ method in the outsider security model. Recall, $\mathcal{A}'$ only knows $(pk_S^{sig}, pk_R^{enc})$, but has access to signcryption and the unsigncryption oracles $\texttt{Signcrypt}$ and $\texttt{Unsigncrypt}$. Let $m_1, \ldots, m_t$ be the queries $\mathcal{A}'$ asks the signcryption oracle and $C_1, \ldots, C_t$ be the corresponding answers. Without loss of generality, we assume that $\mathcal{A}'$ never asks its unsigncryption oracle any query $C'$ which is the same as any one of $C_i$'s returned by the signcryption oracle. Indeed, there is no need for $\mathcal{A}$ to ask such a query since it already knows the answer $m_i$.

Now, we use the standard hybrid argument. Let $Env_0$ denote the usual environment for $\mathcal{A}'$, which honestly answers all the signcryption and unsigncryption queries of $\mathcal{A}'$. Specifically, the signcryption query $m_i$ is answered by computing $\sigma_i \overset{R}{\leftarrow} \texttt{Sign}(sk_S^{sig}, m_i)$ and returning $C_i \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R^{enc}, m_i \| \sigma_i)$. Let $Succ_0(\mathcal{A}')$ be the success probability (i.e., that of breaking sUF-CMA security of the signcryption) of $\mathcal{A}'$ in $Env_0$. Next, we define the following "hybrid" environments $Env_j$, $1 \le j \le t$. Each $Env_j$ is identical to $Env_0$ above, except for one aspect: for the first $j$ queries $m_i$ ($1 \le i \le j$) to the signcryption oracle, instead of returning $C_i \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R^{enc}, m_i \| \sigma_i)$, $Env_i$ returns a random encryption of 0: $C_i \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R^{enc}, 0)$. We let $Succ_j(\mathcal{A}')$ be the success probability of $\mathcal{A}'$ in $Env_j$. Notice, $Env_t$ answers all $t$ queries "incorrectly" (i.e., all signcryption oracle queries are answered with random encryptions of 0).

We make two claims: (1) assuming IND-CCA2 security of $\mathcal{E}$, no PPT adversary $\mathcal{A}'$ can distinguish $Env_{j-1}$ from $Env_j$ with non-negligible probability, for any $1 \le j \le t$, i.e., $|Succ_{j-1}(\mathcal{A}') - Succ_j(\mathcal{A}')| \le negl(k)$, and (2) assuming UF-NMA security of $\mathcal{S}$, $Succ_t(\mathcal{A}') \le negl(k)$, for any PPT $\mathcal{A}'$. Combined, claims (1) and (2) imply our theorem, since $t$ is polynomial and we have

$$\begin{aligned} Succ_0(\mathcal{A}') &\le (Succ_0(\mathcal{A}') - Succ_1(\mathcal{A}')) + \cdots \\ &\qquad \cdots + (Succ_{t-1}(\mathcal{A}') - Succ_t(\mathcal{A}')) + Succ_t(\mathcal{A}') \\ &\le (t+1) \cdot negl(k) \\ &= negl(k) \end{aligned}$$

Proof of Claim (1)
If for some $\mathcal{A}'$, $|Succ_{j-1}(\mathcal{A}') - Succ_j(\mathcal{A}')| > \varepsilon$ for non-negligible $\varepsilon$, then we construct an adversary $\mathcal{A}$ that will break the IND-CCA2 security of $\mathcal{E}$ with probability $\varepsilon$ as follows. Let $(sk_R^{enc}, pk_R^{enc})$ be the key pair for the encryption scheme $\mathcal{E}$. Given the public encryption key $pk_R^{enc}$ and access to the decryption oracle $\texttt{Decrypt}$, $\mathcal{A}$ picks a pair of signing keys $(sk_S^{sig}, pk_S^{sig}) \overset{R}{\leftarrow} \texttt{SigKeyGen}(1^k)$ and gives $(pk_S^{sig}, pk_R^{enc})$ to $\mathcal{A}'$. $\mathcal{A}$ simulates all the unsigncryption queries $C'$ of $\mathcal{A}'$ by using its own decryption oracle on $C'$ to obtain $(m, \sigma)$, then verifying the signature $\sigma$ it gets back, before returning the message $m$ to $\mathcal{A}'$. Simulation of the signcryption oracle is more intricate. $\mathcal{A}$ simulates the answers to the first $(j-1)$ signcryption oracle queries $m_i$ "incorrectly," by returning $C_i \leftarrow \texttt{Encrypt}(pk_R^{enc}, 0)$ to $\mathcal{A}'$ (i.e., returning an encryption of 0). At the $j$th query $m_j$ of $\mathcal{A}'$, $\mathcal{A}$ computes $\sigma_j \overset{R}{\leftarrow} \texttt{Sign}(sk_S^{sig}, m_j)$ and outputs $(0, m_j \| \sigma_j)$ to get the challenge ciphertext $C_j$ (which is an encryption of either 0 or $m_j \| \sigma_j$). $\mathcal{A}$ then gives $C_j$ to $\mathcal{A}'$ as a signcryption of $m_j$. From that point on, all the remaining signcryption queries $m_i$ ($j < i \le t$) are answered "correctly" (i.e., by computing $C_i \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R^{enc}, m_i \| \sigma_i)$ where $\sigma_i \overset{R}{\leftarrow} \texttt{Sign}(sk_S^{sig}, m_i)$).

After $\mathcal{A}'$ returns a candidate forgery $C$, $\mathcal{A}$ checks if $C$ is indeed a valid forgery by (1) checking that $C$ is "new" (i.e., $C$ was never returned to $\mathcal{A}'$ by $\mathcal{A}$ as an answer to a signcryption query in the signcryption oracle simulation) and (2) $C$ is "valid" ($\mathcal{A}$ can check this by using its decryption oracle on $C$ to get the presumed message/signature pair $m \| \sigma$ and verifying that $\sigma$ is a valid signature of $m$). If so, $\mathcal{A}$ guesses that the challenge ciphertext $C_j$ was the encryption of $m_j \| \sigma_j$ (i.e., $\mathcal{A}'$ was run in $Env_{j-1}$), else it guesses that the challenge ciphertext $C_j$ was an encryption of 0. By a method similar to Lemma 1.1, we can show that $\mathcal{A}$'s advantage is $\varepsilon/2$, which is negligible as the encryption scheme is IND-CCA2 secure. However, to complete the proof of claim (1), we also need to check that $\mathcal{A}$ never asked its decryption oracle the challenge ciphertext $C_j$. We assumed that $\mathcal{A}'$ never asks its unsigncryption oracle any query $C_i$ which was returned by the signcryption oracle. Since $\mathcal{A}$ only uses the decryption oracles to answer unsigncryption queries of $\mathcal{A}'$ and to decrypt $C$, this is indeed so.

Proof of Claim (2)
We note that in $Env_t$ (where the signcryption answers are simulated by encrypting 0) the queries to the signcryption oracle are "useless": $\mathcal{A}'$ could have gotten the answers by itself by computing $\texttt{Encrypt}(pk_R^{enc}, 0)$. More formally, assuming $\mathcal{A}'$ forges a new signcryption with probability $\varepsilon$ in $Env_t$, we can build a forger $\mathcal{A}$ for the signature scheme $\mathcal{S}$ that will contradict the UF-NMA security of $\mathcal{S}$. Let $(sk_S^{sig}, pk_S^{sig})$ be the keys of the signature scheme $\mathcal{S}$. Given the public verification key $pk_S^{sig}$, $\mathcal{A}$ picks a pair of encryption keys $(sk_R^{enc}, pk_R^{enc}) \overset{R}{\leftarrow} \texttt{EncKeyGen}(1^k)$ and gives $(pk_S^{sig}, pk_R^{enc})$ to $\mathcal{A}'$ as the public key of the signcryption scheme. From there on, $\mathcal{A}$ simulates the unsigncryption queries $C'$ by computing $m' \| \sigma' = \texttt{Decrypt}(sk_R^{enc}, C')$ and returning $m'$ if $\texttt{Verify}(pk_R^{sig}, m', \sigma') = \top$. It also simulates the signcryption queries by returning $\texttt{Encrypt}(pk_R^{enc}, 0)$. When $\mathcal{A}'$ returns a forged ciphertext $C$, $\mathcal{A}$ outputs the forged message/signature pair $(m, \sigma)$ where $m \| \sigma = \texttt{Decrypt}(sk_R^{enc}, C)$. It is easy to see that $\mathcal{A}$ exactly recreates $Env_t$ and forges a signature only if $\mathcal{A}'$ forges a signcryption. Hence $Succ_t$ is negligible. $\square$

## 2.4 Multi-user Setting

As we have mentioned, the two-user setting provides us with insight into some interesting aspects of signcryption, but one really needs multi-user security for most applications of signcryption. Formal definitions for the security of signcryption in a multi-user setting will be discussed in depth in Chap. 3. In this section, we will provide a brief introduction to multi-user security and the relationship between multi-user security and the generic signcryption constructions.

### *2.4.1 Syntax*

So far we have concentrated on a network of two users: the sender $S$ and the receiver $R$. Once we move to the full-fledged multi-user network, several new concerns arise. First, users must have identities. We denote by $ID_U$ the identity of user $U$. We do not impose any constraints on the identities, other than they should be easily recognizable by everyone in the network and that users can easily obtain the public key $pk_U$ from $ID_U$ (e.g., $ID_U$ could be $pk_U$ or $ID_U$ might enable another user to obtain $pk_U$ from a public-key infrastructure). Next, we change the syntax of the signcryption algorithm $\texttt{Signcrypt}$ to both take and output the identity of the sender and the receiver. Specifically, (1) the signcryption algorithm for user $S$, on input

$(m, ID_{R'})$, uses $pk_{R'}$ to generate $(C, ID_S, ID_{R'})$ and (2) the unsigncryption algorithm for user $R$, on input $(C, ID_{S'}, ID_R)$, uses $pk_{S'}$ and outputs a message $m'$ or the "failure" symbol $\perp$.

### *2.4.2 Security*

To break the outsider security between a pair of designated users $S$ and $R$, $\mathcal{A}$ is assumed to have all the secret keys besides $sk_S$ and $sk_R$ and has access to the signcryption oracle of $S$ (which it can call with *any* $ID_{R'}$ and not just $ID_R$) and the unsigncryption oracle for $R$ (which it can call with *any* $ID_{S'}$ and not just $ID_S$).

To break the sUF-CMA security of the signcryption scheme, the attacker $\mathcal{A}$ has to come up with a "valid" signcryption $(C, ID_S, ID_R)$ of a message $m$ where $\mathcal{A}$ did not receive $(C, ID_S, ID_R)$ as the result of a signcryption oracle query. It is important to note that we do allow the attacker to attempt to generate a forgery by querying the signcryption oracle on $(m, ID_{R'})$ for $ID_R \neq ID_{R'}$ to receive $(C, ID_S, ID_{R'})$ and outputting $(C, ID_S, ID_R)$. This is equivalent to saying that the attacker should not be able to "translate" a signcryption ciphertext intended for $R'$ into a ciphertext intended for $R$.

Similarly, to break IND-CCA2 security of the signcryption scheme, the attacker $\mathcal{A}$ has to generate messages $m_0$ and $m_1$ for which it can distinguish the ciphertext $\texttt{Signcrypt}(m_0, ID_S, ID_R)$ from the ciphertext $\texttt{Signcrypt}(m_1, ID_S, ID_R)$. Of course, given a challenge $(C, ID_S, ID_R)$, $\mathcal{A}$ is disallowed to query the unsigncryption oracle for $R$ on the challenge $(C, ID_S, ID_R)$, although queries of the form $(C, ID_{S'}, ID_R)$, where $ID_{S'} \neq ID_S$, are allowed.

We define insider security in an analogous manner. The only difference is that in addition to all the information given to the adversary in the outsider security model, the adversary is given the receiver's secret key, $sk_R$, when attacking authenticity (i.e., $sk_S$ is the only secret that is not given to the adversary in this case) and the sender's secret key, $sk_S$, when attacking privacy (i.e., $sk_R$ is the only secret that is not given to the adversary in this case).

### *2.4.3 Extending Signcryption*

We can see that the signcryption algorithms that are built by generic composition of encryption and signature schemes (i.e., $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$) are not secure in the multi-user setting. If the $\mathcal{E}t\mathcal{S}$ method is used in the multi-user setting, then the adversary $\mathcal{A}$ can easily break the CCA2 security, even in the outsider model. Indeed, given the challenge $C = (c, \sigma, ID_S, ID_R)$, where $c \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R, m_b)$ and $\sigma \overset{R}{\leftarrow} \texttt{Sign}(sk_S, c)$, $\mathcal{A}$ can replace the sender's signature with its own by computing $C' = (c, \sigma', ID_{S'}, ID_R)$, where $\sigma' \overset{R}{\leftarrow} \texttt{Sign}(sk_{S'}, c)$. If $\mathcal{A}$ queries the unsigncryption oracle on $C'$ then the oracle will respond with $m_b$ and $\mathcal{A}$ can trivially break the IND-CCA2 security of the scheme. A similar attack on authenticity holds for the $\mathcal{StE}$ scheme. In the $\mathcal{StE}$ scheme, the adversary $\mathcal{A}$ can easily break the sUF-CMA security in the outsider model. It can ask $S$ to signcrypt a message $m$ for $R'$ and get $C = (\texttt{Encrypt}(pk_{R'}, m\|\sigma), ID_S, ID_{R'})$, where $\sigma \overset{R}{\leftarrow} \texttt{Sign}(pk_S, m)$. Then, it can recover $m\|\sigma$ using $sk_{R'}$ and forge the signcryption ciphertext $C' = (\texttt{Encrypt}(pk_R, m\|\sigma), ID_S, ID_R)$.

The generic composition methods suffer from the above types of attacks in the multi-user setting, because the signature and encryption used in the signcryption can easily be separated and are not "bound together" with the proper identities of the sender and receiver (unlike the two-user setting or the symmetric setting). The adversary can easily replace the signature or encryption with its own signature or encryption. We show how to fix this problem by "binding together" the signature and encryption used in the signcryption with the proper identities of the sender and receiver. The following rules can effectively bind the encryption and signature with proper identities of the sender and receiver and hence can be used to make the signcryption schemes built by generic composition secure in the multi-user setting (i.e. withstand above types of attacks).

1. Whenever *encrypting* something, include the identity of the *sender* $ID_S$ together with the encrypted message.
2. Whenever *signing* something, include the identity of the *receiver* $ID_R$ together with the signed message.
3. On the receiving side, whenever the identity of either the sender or the receiver does not match what is expected, output $\perp$.

Hence, we get the following new analogs for the $\mathcal{EtS}$ and $\mathcal{StE}$ schemes:

- The $\mathcal{EtS}$ signcryption scheme returns the signcryption ciphertext $(c, \sigma, ID_S, ID_R)$ where $c \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R, m\|ID_S)$ and $\sigma \overset{R}{\leftarrow} \texttt{Sign}(sk_S, c\|ID_R)$.
- The $\mathcal{StE}$ signcryption scheme returns the signcryption ciphertext $(c, ID_S, ID_R)$ where $c \overset{R}{\leftarrow} \texttt{Encrypt}(pk_R, m\|\sigma\|ID_S)$ and $\sigma \overset{R}{\leftarrow} \texttt{Sign}(sk_S, m\|ID_R)$.

For both schemes, the unsigncryption algorithms work in the obvious manner. Intuitively, it is easy to see that the above rules "bind" the encryption and signature used in the signcryption with proper identities of the sender and receiver, because it includes the intended sender and receiver identities in the ciphertext.

However, it is important to ensure that the identities cannot be tampered within the ciphertext itself. If the encryption scheme used in the signcryption is malleable (i.e., underlying plaintext can be modified without being detected), the adversary may be able to modify the identities in the ciphertext which makes having the identities moot. For example, in the $\mathcal{EtS}$ method, if the underlying encryption scheme is only IND-CPA secure, the adversary may be able to modify the ciphertext to replace the sender's identity with its own and strip off the sender's signature and replace it with its own signature and identity. Hence, it is important to assume that the underlying encryption scheme is non-malleable (or CCA2 secure) for the signcryption scheme built by the $\mathcal{EtS}$ method to be secure even in the outsider model. This is different from the result in the two-user setting, where the encryption scheme can be just IND-CPA secure for the signcryption built from the $\mathcal{E}t\mathcal{S}$ method to be (IND-CCA2 and sUF-CMA) secure in the outsider model. This tells us that the security proven in the two-user setting does *not* automatically translate into security in the multi-user setting even if we follow above rules to “bind” the signature and encryption with identities. In general, when analyzing the security of the signcryption scheme built from the generic composition methods following the above rules in the multi-user setting, the assumptions for the underlying encryption and signature schemes should be “strong enough” (i.e., IND-CCA2 or sUF-CMA secure) so that the identities bound to the signature and encryption cannot be altered.