On the Security of Joint Signature and Encryption

Jee Hea An[*] Yevgeniy Dodis[†] Tal Rabin[‡]

June 17, 2002

Abstract

We formally study the notion of a joint signature and encryption in the public-key setting. We refer to this primitive as *signcryption*, adapting the terminology of [36]. We present two definitions for the security of signcryption depending on whether the adversary is an outsider or a legal user of the system. We then examine generic sequential composition methods of building signcryption from a signature and an encryption scheme. In particular, we show that the classical "encrypt-then-sign" ($\mathcal{E}t\mathcal{S}$) and "sign-then-encrypt" ($\mathcal{S}t\mathcal{E}$) methods — when modeled "properly" — are both *secure* composition methods in the public-key setting.

We also present a new composition method which we call "commit-then-encrypt-and-sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$). Unlike the generic sequential composition methods, $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ applies the expensive signature and encryption operations *in parallel*, which could imply a gain in efficiency over the $\mathcal{S}t\mathcal{E}$ and $\mathcal{E}t\mathcal{S}$ schemes. We also show that the new $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ method elegantly combines with the recent "hash-sign-switch" technique of [31], leading to efficient *on-line/off-line* signcryption.

Finally and of independent interest, we discuss the *definitional* inadequacy of the standard notion of chosen ciphertext (CCA2) security. We suggest a natural and very slight relaxation of CCA2-security, which we call generalized CCA2-security (gCCA2). We show that gCCA2-security suffices for all known uses of CCA2-secure encryption, while no longer suffering from the definitional shortcomings of the latter.

[*]SoftMax Inc., San Diego, USA. (Work done while at UCSD). Email: jeehea@cs.ucsd.edu

[†]Department of Computer Science, New York University, 251 Mercer Street, New York, NY 10012, USA. Email: dodis@cs.nyu.edu

[‡]IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598, USA. Email: talr@watson.ibm.com.

1 Introduction

Signcryption. Encryption and signature schemes are fundamental cryptographic tools for providing privacy and authenticity, respectively, in the public-key setting. Until very recently, they have been viewed as important but *distinct* basic building blocks of various cryptographic systems, and have been designed and analyzed separately. The separation between the two operations can be seen as a natural one as encryption is aimed at providing privacy while signatures are used to enable authentication, and these are two fundamentally different security goals. Yet clearly, there are many settings where both are needed, perhaps the most basic one is in secure e-mailing, where each message should be authenticated and encrypted. A straightforward solution to offering simultaneously both privacy and authenticity might be to compose the known solutions of each of the two components. But given that the combination of the two security goals is such a common and, in fact, basic task, it stands to reason that a tailored solution for the combination should be given. Indeed, a cryptographic tool providing both authenticity and privacy has usually been called an *authenticated encryption*, but was mainly studied in the symmetric setting [6, 5, 23]. This paper will concentrate on the corresponding study in the public key setting, and will use the term *signcryption* to refer to a "joint signature and encryption". We remark that this term was originally introduced and studied by Zheng [36] with the primary goal of reaching greater efficiency than when carrying out the signature and encryption operations separately. As we will argue shortly, efficiency is only one (albeit important) concern when designing a secure joint signature and encryption. Therefore, we will use the term "signcryption" for *any* scheme achieving both privacy and authenticity in the public key setting, irrespective of its performance, as long as it satisfies a formal definition of security we develop in this paper. Indeed, despite presenting some security arguments, most of the initial work on signcryption [36, 37, 27, 20] lacked formal definitions and analysis. This paper will provide such a formal treatment, as well as give new general constructions of signcryption.

Signcryption as a primitive? Before devoting time to the definition and design of (additional) signcryption schemes one must ask if there is a need for defining signcryption as a separate primitive. Indeed, maybe one should forgo this notion and always use a simple composition of a signature and encryption? Though we show in the following that these compositions, in many instances, yield the desired properties, we still claim that a separate notion of signcryption is extremely useful. This is due to several reasons. First, under certain definitions of security (i.e., so called CCA2-security as explained in Section 8), the straightforward composition of a secure signature and encryption does *not* necessarily yield a secure signcryption. Second, as we show in Section 3, there are quite subtle and "non-obvious" issues with respect to signcryption — especially in the public-key setting — which need to be captured in a formal definition. Third, there are other interesting constructions for signcryption which do not follow the paradigm of sequentially composing signatures and encryption. Fourth, designing tailored solutions might yield efficiency (which was the original motivation of Zheng [36]). Finally, the usage of signcryption as a *primitive* might conceptually simplify the design of complex protocols which require both privacy and authenticity (e.g., some key exchange protocols; see [32]).

Summarizing the above discussion, we believe that the study of signcryption *as a primitive* is important and can lead to very useful, general as well as specific, paradigms for achieving privacy and authenticity at the same time.

Our Results. This paper provides a formal treatment of signcryption and analyzes several general constructions for this primitive. In particular, we note that the problem of defining signcryption in the public key setting is more involved than the corresponding task in the symmetric setting studied by [5, 23], due to the asymmetric nature of the former. For example, full-fledged signcryption needs to be defined in the *multi-user* setting, where subtle 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. Luckily, we show that for all the generic schemes we address in this paper,[1] it suffices to analyze their security in the two-user setting, by giving a "semi-generic" generic transformation to the multi-user setting.[2]

We give two definitions for security of signcryption depending on whether the adversary is an outsider or a legal user of the network (i.e., either the sender or the receiver). In both of these settings, we show that the common "encrypt-then-sign" ($\mathcal{E}t\mathcal{S}$) and "sign-then-encrypt" ($\mathcal{S}t\mathcal{E}$) methods in fact yield a *secure* signcryption, provided an *appropriate* definition of security is used. Moreover, when the adversary is an outsider, these composition methods can actually provide *stronger* privacy or authenticity properties for the resulting signcryption scheme than the assumed security properties on the base encryption or signature scheme. Specifically, the security of the base signature scheme can help amplify the privacy of $\mathcal{E}t\mathcal{S}$, while the security of the base encryption scheme can do the same to the authenticity of $\mathcal{S}t\mathcal{E}$. We remark that these possibly "expected" results are nevertheless somewhat surprising in light of recent "negative" indications from the symmetric setting [5, 23], and illustrate the need for rigorous definitions for security of signcryption.

In addition, we present a novel construction of signcryption, which we call "commit-then-encrypt-and-sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$). Our scheme is a general way to construct signcryption from any signature and encryption schemes, while utilizing in addition a commitment scheme. This method is quite different from the obvious sequential composition paradigm. Moreover, unlike the previous sequential methods, the $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ method applies the expensive signature and encryption operations *in parallel*, which could imply a gain in efficiency. We also show that our construction naturally leads to a very efficient way to implement *off-line signcryption*, where the sender can prepare most of the authenticated ciphertext in advance (i.e., without knowing the plaintext) and perform very little on-line computation.

Finally and of independent interest, we discuss the *definitional* inadequacy of the standard notion of chosen ciphertext (CCA2) security [14, 4]. Motivated by our applications to signcryption, we show that the notion of CCA2-security is syntactically ill-defined, and leads to artificial examples of "intuitively CCA2-secure" schemes which do not meet the formal definition (such observations were also made by [8, 9]). We suggest a natural and very slight relaxation of CCA2-security, which we call *generalized* CCA2-*security* (gCCA2). We show that gCCA2-security suffices for all known uses of CCA2-secure encryption, while no longer suffering from the definitional shortcomings of the latter.

Related Work. The initial works on signcryption [36, 37, 27, 20] designed several signcryption schemes, whose "security" was informally based on various number-theoretic assumptions. Only recently (and independently of our work) Baek et al. [3] showed that the original scheme of Zheng [36] (based on shortened ElGamal signatures) can be shown secure in the random oracle model under the gap Diffie-Hellman assumption.

We also mention the works of [35, 30], which used Schnorr signature to amplify the security of ElGamal encryption to withstand a chosen ciphertext attack. However, the above works concentrate on providing privacy, and do not provide authenticity, as required by our notion of signcryption.

Recently, much work has been done about authenticated encryption in the symmetric (private-key) setting. The first formalizations of authenticated encryption in the symmetric setting were done by [22, 6, 5]. The works of [5, 23] discuss the security of generic composition methods of a (symmetric) encryption and a message authentication code (MAC). In particular, a lot of emphasis in these works is given to the study of sufficient conditions under which a given composition method can *amplify* (rather

[1] Specifically, those that are designed from a general signature and encryption scheme (rather than in an "ad-hoc" manner from a specific number-theoretic assumption).

[2] It is an interesting open question if one can reduce multi-user security to the two-user security for *arbitrary* signcryption schemes, and not just the ones build from general signature and encryption.

than merely preserve) the privacy property of a given composition method from the chosen plaintext (CPA) to the chosen ciphertext (CCA2) level. From this perspective, the "encrypt-then-mac" method — which always achieves such an amplification due to a "strongly unforgeable" MAC — was found generically preferable to the "mac-then-encrypt" method, which does so only in specific (albeit very useful) cases [23]. In contrast, An and Bellare [1] study a symmetric question of under which conditions a "good" privacy property on the base encryption scheme can help amplify the authenticity property in the "mac-then-encrypt" (or "encrypt-with-redundancy") method. On a positive side, they found that chosen ciphertext security on the base encryption scheme is indeed sufficient for that purpose. As we shall see in Section 4, all these results are very related to our results about "sign-then-encrypt" and "encrypt-then-sign" methods for signcryption when the adversary is an "outsider".

Another related paradigm for building authenticated encryption is the "encode-then-encipher" method of [6]: add randomness and redundancy, and then encipher (i.e., apply a pseudorandom permutation) rather than encrypt. Even though a strong pseudorandom permutation is often more expensive than encryption, [6] shows that very simple *public* redundancy functions are sufficient — in contrast to the "encrypt-with-redundancy" method, where no public redundancy can work [1].

Finally, we mention recently designed modes of operations for block ciphers that achieve both privacy and authenticity in the symmetric setting: RFC mode of [22], IACBC and IAPM modes of [21], OCB mode of [29], and SNCBC mode of [1].

2 Definitions

In this section we briefly review the (public-key) notions of encryption, signature and commitment schemes. In addition, we present our extended definition for CCA2.

2.1 Encryption

SYNTAX. An encryption scheme consists of three algorithms: $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. $\mathsf{Enc\text{-}Gen}(1^k)$, where k is the security parameter, outputs a pair of keys $(\mathsf{EK}, \mathsf{DK})$. EK is the encryption key, which is made public, and DK is the decryption key which is kept secret. The randomized encryption algorithm Enc takes as input a key EK and a message m from the associated message space $\mathcal{M}$, and internally flips some coins and outputs a ciphertext e; we write $e \leftarrow \mathsf{Enc}_{\mathsf{EK}}(m)$. For brevity, we will usually omit EK and write $e \leftarrow \mathsf{Enc}(m)$. The deterministic decryption algorithm Dec takes as input the ciphertext e, the secret key DK, and outputs some message $m \in \mathcal{M}$, or $\perp$ in case e was "invalid". We write $m \leftarrow \mathsf{Dec}(e)$ (again, omitting DK). We require that $\mathsf{Dec}(\mathsf{Enc}(m)) = m$, for any $m \in \mathcal{M}$.

SECURITY OF ENCRYPTION. When addressing the security of the schemes, we deal with two issues: what we want to achieve (security goal) and what are the capabilities of the adversary (attack model). In this paper we will talk about the most common security goal: indistinguishability of ciphertexts [17], which we will denote by IND. A related notion of *non-malleability* will be briefly discussed in Section 8.

Intuitively, indistinguishability means that given a randomly selected public key, no PPT (probabilistic polynomial time) adversary $\mathcal{A}$ can distinguish encryptions of any two messages m_0, m_1 chosen by $\mathcal{A}$: $\mathsf{Enc}(m_0) \approx \mathsf{Enc}(m_1)$. Formally, we require that for any PPT $\mathcal{A}$, which runs in two stages, find and guess, we have

$$\Pr\left[\ b = \tilde{b}\ \middle|\ \begin{array}{c} (\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}Gen}(1^k),\ (m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathsf{EK}, \mathsf{find}), \\ b \overset{R}{\leftarrow} \{0,1\},\ e \leftarrow \mathsf{Enc}_{\mathsf{EK}}(m_b),\ \tilde{b} \leftarrow \mathcal{A}(e, \alpha, \mathsf{guess}) \end{array}\ \right] \le \frac{1}{2} + \mathsf{negl}(k)$$

Here and elsewhere $\mathsf{negl}(k)$ is some negligible function in the security parameter k, and α is some internal state information $\mathcal{A}$ saves and uses in the two stages.

We now turn to the second issue of security of encryption — the attack model. We consider three types of attack: CPA, CCA1 and CCA2. Under the *chosen plaintext* (or CPA) attack, the adversary is not given any extra capabilities other than encrypting messages using the public encryption key. A more powerful type of *chosen ciphertext* attack gives $\mathcal{A}$ access to the decryption oracle, namely the ability to decrypt arbitrary ciphertexts of its choice. The first of this type of attack is the *lunch-time* (CCA1) attack [28], which gives access only in the find stage (i.e., before the challenge ciphertext e is given). The second is CCA2 on which we elaborate in the following.

CCA2 ATTACKS. The *adaptive chosen ciphertext attack* [14] (CCA2) gives access to the decryption oracle in the guess stage as well. As stated, the CCA2 attack does not make sense since $\mathcal{A}$ can simply ask to decrypt the challenge e. Therefore, we need to restrict the class of ciphertexts e' that $\mathcal{A}$ can give to the decryption oracle in the guess stage. The minimal restriction is to have $e' \neq e$, which is the way the CCA2 attack is usually defined. As we will argue in Section 8, stopping at this minimal (and needed) restriction in turn restricts the class of encryption schemes that we intuitively view as being "secure". In particular, it is not robust to syntactic changes in the encryption (e.g., appending a harmless random bit to a secure encryption suddenly makes it "insecure" against CCA2). Leaving further discussion to Section 8, we now define a special case of the CCA2 attack which does not suffer from the above syntactic limitations and suffices for all the uses of the CCA2-secure encryption we are aware of.

We first generalize the CCA2 attack with respect to some equivalence relation $\mathcal{R}(\cdot,\cdot)$ on the ciphertexts. $\mathcal{R}$ is defined as part of the encryption scheme, it can depend on the public key EK, but must have the following property: if $\mathcal{R}(e_1, e_2) = \mathit{true} \Rightarrow \mathsf{Dec}(e_1) = \mathsf{Dec}(e_2)$. We call such $\mathcal{R}$ *decryption-respecting*. Now $\mathcal{A}$ is forbidden to ask any e' equivalent to e, i.e. $\mathcal{R}(e, e') = \mathit{true}$. Since $\mathcal{R}$ is reflexive, this at least rules out e, and since $\mathcal{R}$ is decryption-respecting, it only restricts ciphertexts that decrypt to the same value as the decryption of e (i.e. m_b). We note that the usual CCA2 attack corresponds to the equality relation. Now we say that the encryption scheme is secure against *generalized* CCA2 (or gCCA2) if there *exists* some efficient decryption-respecting relation $\mathcal{R}$ w.r.t. which it is CCA2-secure. For example, appending a harmless bit to gCCA2-secure encryption or doing other easily recognizable manipulation still leaves it gCCA2-secure.

We remark that the notion of gCCA2-security was recently proposed in [33] (under the name *benign malleability*) for the ISO public key encryption standard. In the private-key setting, [23] uses equivalences relations to define "loose ciphertext unforgeability".

2.2 Signatures

SYNTAX. A signature scheme consists of three algorithms: $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$. $\mathsf{Sig\text{-}Gen}(1^k)$, where k is the security parameter, outputs a pair of keys (SK, VK). SK is the signing key, which is kept secret, and VK is the verification key which is made public. The randomized signing algorithm Sig takes as input a key SK and a message m from the associated message space $\mathcal{M}$, internally flips some coins and outputs a signature s; we write $s \leftarrow \mathsf{Sig}_{\mathsf{SK}}(m)$. We will usually omit SK and write $s \leftarrow \mathsf{Sig}(m)$. Wlog, we will assume that the message m can be determined from the signature s (e.g., is part of it), and write $m = \mathsf{Msg}(s)$ to denote the message whose signature is s. The deterministic verification algorithm Ver takes as input the signature s, the public key VK, and outputs the answer a which is either succeed (signature is valid) or fail (signature is invalid). We write $a \leftarrow \mathsf{Ver}(s)$ (again, omitting VK). We require that $\mathsf{Ver}(\mathsf{Sig}(m)) = \mathsf{succeed}$, for any $m \in \mathcal{M}$.

SECURITY OF SIGNATURES. As with the encryption, the security of signatures addresses two issues: what we want to achieve (security goal) and what are the capabilities of the adversary (attack model). In this paper we will talk about the the most common security goal: *existential unforgeability* [18], denoted by UF. This means that any PPT adversary $\mathcal{A}$ should have a negligible probability of generating a valid signature of a "new" message. To clarify the meaning of "new", we will consider the following two attack models. In the *no message attack* (NMA), $\mathcal{A}$ gets no help besides VK. In the *chosen message attack* (CMA), in addition to VK, the adversary $\mathcal{A}$ gets full access to the signing oracle Sig, i.e. $\mathcal{A}$ is allowed to query the signing oracle to obtain valid signatures $s_1, \ldots, s_n$ of arbitrary messages $m_1, \ldots, m_n$ adaptively chosen by $\mathcal{A}$ (notice, NMA corresponds to $n = 0$). Naturally, $\mathcal{A}$ is considered successful only if it forges a valid signature s of a message m not queried to signing oracle: $m \notin \{m_1 \ldots m_n\}$. We denote the resulting security notions by $\mathsf{UF\text{-}NMA}$ and $\mathsf{UF\text{-}CMA}$, respectively.

We also mention a slightly stronger type of unforgeability called *strong unforgeability*, denoted sUF. Here $\mathcal{A}$ should not only be unable to generate a signature of a "new" message, but also be unable to generate even a different signature of an already signed message, i.e. $s \notin \{s_1, \ldots, s_n\}$. This only makes sense for the CMA attack, and results in a security notion we denote by $\mathsf{sUF\text{-}CMA}$.

2.3 Commitment

Syntax. A (non-interactive) commitment scheme consists of three algorithms: $\mathcal{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$. The setup algorithm $\mathsf{Setup}(1^k)$, where k is the security parameter, outputs a public commitment key CK (possibly empty, but usually consisting of public parameters for the commitment scheme). Given a message m from the associated message space $\mathcal{M}$ (e.g., $\{0,1\}^k$), $\mathsf{Commit}_{\mathsf{CK}}(m; r)$ (computed using the public key CK and additional randomness r) produces a commitment pair (c, d), where c is the *commitment* to m and d is the *decommitment*. We will usually omit CK and write $(c, d) \leftarrow \mathsf{Commit}(m)$. Sometimes we will write $c(m)$ (resp. $d(m)$) to denote the commitment (resp. decommitment) part of a randomly generated (c, d). The last (deterministic) algorithm $\mathsf{Open}_{\mathsf{CK}}(c, d)$ outputs m if (c, d) is a *valid* pair for m (i.e. could have been generated by $\mathsf{Commit}(m)$), or $\perp$ otherwise. We require that $\mathsf{Open}(\mathsf{Commit}(m)) = m$ for any $m \in \mathcal{M}$.

Security of Commitment. Regular commitment schemes have two security properties:

Hiding. No PPT adversary can distinguish the commitments to any two message of its choice: $c(m_1) \approx c(m_2)$. That is, $c(m)$ reveals "no information" about m. Formally, for any PPT $\mathcal{A}$ which runs in two stages, find and guess, we have

$$\Pr\left[b = \tilde{b} \;\middle|\; \begin{array}{c} \mathsf{CK} \leftarrow \mathsf{Setup}(1^k), (m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathsf{CK}, \mathsf{find}), \\ b \stackrel{R}{\leftarrow} \{0,1\}, (c, d) \leftarrow \mathsf{Commit}_{\mathsf{CK}}(m_b), \tilde{b} \leftarrow \mathcal{A}(c;\ \alpha, \mathsf{guess}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(k)$$

Binding. Having the knowledge of CK, it is computationally hard for the adversary $\mathcal{A}$ to come up with c, d, d' such that (c, d) and (c, d') are valid commitment pairs for m and m', but $m \neq m'$ (such a triple c, d, d' is said to cause a *collision*). That is, $\mathcal{A}$ cannot find a value c which it can open in two different ways.

Relaxed commitments. We will also consider *relaxed* commitment schemes, where the (strict) binding property above is replaced by the **Relaxed Binding** property: for any PPT adversary $\mathcal{A}$, having the knowledge of CK, it is computationally hard for $\mathcal{A}$ to come up with a message m, such that when $(c, d) \leftarrow \mathsf{Commit}(m)$ is generated, $\mathcal{A}(c, d, \mathsf{CK})$ produces, with non-negligible probability, a value d' such that (c, d') is a valid commitment to some $m' \neq m$. Namely, $\mathcal{A}$ cannot find a collision using a *randomly generated* $c(m)$, even for m of its choice.

To justify this distinction, first recall the concepts of collision-resistant hash function (CRHF) families and universal one-way hash function (UOWHF) families. For both concepts, it is hard to find a colliding pair $x \neq x'$ such that $H(x) = H(x')$, where H is a function randomly chosen from the corresponding family. However, with CRHF, we first select the function H, and for UOWHF the adversary has to select x before H is given to it. By the result of Simon [34], UOWHF's are strictly weaker primitive than CRHF (in particular, they can be built from regular one-way functions [25]). We note two classical results about (regular) commitment schemes: the construction of such a scheme by [11, 19], and the folklore "hash-then-commit" paradigm (used for committing to long messages by hashing them first). Both of these results require the use of CRHF's, and it is easy to see that UOWHF's are not sufficient to ensure (strict) binding for either one of them. On the other hand, it is not very hard to see that UOWHF's suffice to ensure relaxed binding in both cases. Hence, basing some construction on relaxed commitments (as we will do in Section 5) has its merits over regular commitments.

TRAPDOOR COMMITMENTS. We also define a very useful class of commitment schemes, known as (non-interactive) *trapdoor commitments* [7] or *chameleon hash functions* [24]. In these schemes the setup algorithm $\mathsf{Setup}(1^k)$ outputs a pair of keys $(\mathsf{CK}, \mathsf{TK})$. That is, in addition to the public commitment key CK, it also produces a *trapdoor* key TK. Like regular commitments, trapdoor commitments satisfy the hiding property and (possibly relaxed) binding properties. Additionally, they have an efficient switching algorithm Switch, which allows one to find arbitrary collisions using the trapdoor key TK.

Trapdoor Collisions. Given any commitment pair (c, d) to some message m and *any* message m', $\mathsf{Switch}_{\mathsf{TK}}((c, d), m')$ outputs a valid commitment pair (c, d') to m' (note, c is the same!). Moreover, having the knowledge of CK, it is computationally hard to come up with two messages m, m' such that the adversary can distinguish $\mathsf{Commit}_{\mathsf{CK}}(m')$ from $\mathsf{Switch}_{\mathsf{TK}}(\mathsf{Commit}_{\mathsf{CK}}(m), m')$. Namely, a true commitment pair for m' looks the same as a faked commitment pair for m' (obtained from a random pair for m).

We note that the trapdoor collisions property is much stronger (and easily implies) the hiding property (since the switching algorithm does not change $c(m)$). Moreover, the hiding property is *information-theoretic*. We also note that very efficient trapdoor commitment schemes exist based on factoring [24, 31] or discrete log [24, 7]. In particular, the switching function requires just one modulo addition and one modulo multiplication for the discrete log based solution. Less efficient constructions based on more general assumptions are known as well [24].

3 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) 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 (i.e., 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. This suggests that signcryption should be defined in the *multi-user* setting. Luckily, however, all our signcryption schemes will be composed from general signature and encryption schemes as building blocks. And we will show that for such schemes there is a "generic transformation" (see Section 7) from two- to multi-user security. In other words, *for our purposes* we can first define and study the crucial properties of signcryption in the stand-alone two-user setting, and then add identities to our definitions and constructions to achieve the full-fledged multi-user security.

3.1 Some Philosophy: Two- vs. Multi-user Security

As we shall see, first starting from the two-user setting will have certain advantages. For one thing, there are quite a few subtle issues with defining the security of signcryption even in the simpler two-user setting, which makes it a non-trivial starting point before moving to the more complex setting. Secondly, we already mentioned that it will make our presentation much clearer at no cost, since our specific schemes can be easily extended to the multi-user setting. More importantly, however, there is one more conceptual reason for distinguishing two-user and multi-user settings.

Essentially, it turns out that the major additional issue addressed in the multi-user (but not in the two-user) setting is that of *identity fraud*. Namely, since identities have to be explicitly included with each signcryption in the multi-user setting, it should be hard to tamper with these identities without the recipient noticing the difference. For example, if Alice sends some message to Bob, the adversary should not be able to modify the signcryption so that Bob thinks that the message (possibly unknown to the adversary) actually came from some other party Charlie. Similarly, the adversary should not be able to convince Charlie that Alice sent this message to him rather than to Bob. On the other hand, security in the two-user setting more or less ensures that there are no other attacks on the signcryption beside trying to tamper with the identities. In particular, if the identity of the sender and the recipient are known a-priori, the basic signcryption is completely private (adversary cannot understand the message), and authentic (adversary cannot convince Bob that Alice send him some message that she did not sent). Put differently, two-user security ignores the issue of identities and guarantees there are no weaknesses in the way the signcryption *itself* is built.

We feel that this separation is quite important both for our understanding of what "secure signcryption" is, as well as how to design and analyze signcryption schemes. Consider, for example, a very simple $\mathcal{E}t\mathcal{S}$ scheme, where the sender Alice first encrypts the message with the recipient's Bob public key, and then signs the resulting ciphertext (with her signing key). As we will show later, $\mathcal{E}t\mathcal{S}$ is "secure" in the two-user setting. On the other hand, in the multi-user setting the adversary Eve can easily "strip" Alice's "outside" signature and replace it with his own signature, convincing Bob that the message (possibly unknown to Eve!) came from him. Notice that this attack did not really expose any weaknesses in the signature or the encryption scheme: and, indeed, it *could not* since the scheme is secure in the two-user setting. Of course, we will show in Section 7 a very simple fix to the $\mathcal{E}t\mathcal{S}$ scheme above to make it secure even in the multi-user setting. However, the above example illustrates the usefulness of our separation: treating $\mathcal{E}t\mathcal{S}$ in the two-user setting allows for a very clean justification that the *message* is securely hidden inside the signed ciphertext, while starting right away with multi-user setting makes the scheme more complex, the argument less elegant, and the intuition less transparent.

Of course, in general one certainly needs to prove the multi-user security of the proposed scheme, since identity fraud protection is very important. We therefore suggests the following methodology. First one should to design a signcryption scheme which is secure in the two-user setting, which typically provides a good intuition behind the basic scheme. Ones this is done, one should see what (if anything) "goes wrong" in the multi-user setting, and see if some simple measures can "fix" the problem. Of course, this methodology is not universal, but it certainly works in our case and makes our presentation much more transparent. Finally, we will see that there are quite a few subtle issues with defining the security of signcryption even in the much simpler two-user setting, which additionally justifies our two-level approach.

Summarizing the above discussion, for now we concentrate on a simple two-user setting, postponing the extension to multi-user setting to Section 7.

3.2 Two Security Notions in the Two-User Setting

Syntax. A signcryption scheme $\mathcal{SC}$ consists of three algorithms: $\mathcal{SC} = (\mathsf{Gen}, \mathsf{SigEnc}, \mathsf{VerDec})$. The algorithm $\mathsf{Gen}(1^k)$, where k is the security parameter, outputs a pair of keys $(\mathsf{SDK}, \mathsf{VEK})$. SDK is the user's sign/decrypt key, which is kept secret, and VEK the user's verify/encrypt key, which is made public. Note, that in the signcryption setting *all* participating parties need to invoke Gen. For a user P, denote its keys by SDK_P and VEK_P. The randomized *signcryption* (sign/encrypt) algorithm SigEnc takes as input the sender S's secret key SDK_S and the receiver R's public key VEK_R and a message m from the associated message space $\mathcal{M}$, and internally flips some coins and outputs a signcryption (ciphertext) u; we write $u \leftarrow \mathsf{SigEnc}(m)$ (omitting SDK_S, VEK_R). The deterministic *de-signcryption* (verify/decrypt) algorithm VerDec takes as input the signcryption (ciphertext) e, the receiver R's secret key SDK_R and the sender S's public key VEK_S, and outputs $m \in \mathcal{M} \cup \{\perp\}$, where $\perp$ indicates that the message was not encrypted or signed properly. We write $m \leftarrow \mathsf{VerDec}(u)$ (again, omitting the keys). We require that $\mathsf{VerDec}(\mathsf{SigEnc}(m)) = m$, for any $m \in \mathcal{M}$.

Security of Signcryption. Fix the sender S and the receiver R. Intuitively, we would like to say that S's authenticity is protected, and R's privacy is protected. We will give two formalizations of this intuition. The first one assumes that the adversary $\mathcal{A}$ is an outsider who only knows the public information $pub = (\mathsf{VEK}_R, \mathsf{VEK}_S)$. We call such security *Outsider security*. The second, stronger notion, protects S's authenticity even against R, and R's privacy even against S. Put in other words, it assumes that the adversary $\mathcal{A}$ is a legal user of the system. We call such security *Insider security*.

Outsider Security. We define it against the strongest security notions on the signature (analogs of UF-CMA or sUF-CMA) and encryption (analogs of IND-gCCA2 or IND-CCA2), and weaker notions could easily be defined as well. We assume that the adversary $\mathcal{A}$ has the public information $pub = (\mathsf{VEK}_S, \mathsf{VEK}_R)$. It also has oracle access to the functionalities of both S and R. Specifically, it can mount a chosen message attack on S by asking S to produce signcryption u 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 R by giving R any candidate signcryption u and receiving back the message m (where m could be $\perp$), i.e. $\mathcal{A}$ has access to the *de-signcryption oracle*. Notice, $\mathcal{A}$ cannot by itself run either the signcryption or the de-signcryption oracles due to the lack of corresponding secret keys SDK_S and SDK_R.

To break the UF-CMA security of the signcryption scheme, $\mathcal{A}$ has to come up with a *valid* signcryption u of a "new" message m, which it did not ask S to signcrypt earlier (notice, $\mathcal{A}$ is not required to "know" m when producing u). The scheme is *Outsider-secure* in the UF-CMA sense if any PPT $\mathcal{A}$ has a negligible chance of succeeding. (For sUF-CMA, $\mathcal{A}$ only has to produce u which was not returned by S earlier.)

To break the indistinguishability of the signcryption scheme, $\mathcal{A}$ has to come up with two messages m_0 and m_1. One of these will be signcrypted at random, the corresponding signcryption u will be given to $\mathcal{A}$, and $\mathcal{A}$ has to guess which message was signcrypted. To succeed in the CCA2 attack, $\mathcal{A}$ is only disallowed to ask R to de-signcrypt the challenge u. For gCCA2 attack, similarly to the encryption scenario, we first define CCA2 attack against a given efficient decryption-respecting relation $\mathcal{R}$ (which could depend on $pub = (\mathsf{VEK}_R, \mathsf{VEK}_S)$ but not on any of the secret keys). As before, decryption-respecting means that $\mathcal{R}(u, u') = true \Rightarrow \mathsf{VerDec}(u) = \mathsf{VerDec}(u')$. Thus, CCA2 attack w.r.t. $\mathcal{R}$ disallows $\mathcal{A}$ to de-signcrypt any u' equivalent to the challenge u. Now, for Outsider-security against CCA2 w.r.t. $\mathcal{R}$, we require $\Pr[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(k)$. Finally, the scheme is *Outsider-secure* in the IND-gCCA2 sense if it is Outsider-secure against CCA2 w.r.t. *some* efficient decryption-respecting $\mathcal{R}$.

Insider Security. We could define Insider security in a similar manner by defining the capabilities of $\mathcal{A}$ and its goals. However, it is much easier to use *already existing* security notions for signature and encryption schemes. Moreover, this will capture the intuition that "signcryption = signature + encryption". More precisely, given any signcryption scheme $\mathcal{SC} = (\mathsf{Gen}, \mathsf{SigEnc}, \mathsf{VerDec})$, we define the corresponding *induced* signature scheme $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ and encryption scheme $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$.

- **Signature $\mathcal{S}$.** The generation algorithm Sig-Gen runs $\mathsf{Gen}(1^k)$ twice to produce two key pairs $(\mathsf{SDK}_S, \mathsf{VEK}_S)$ and $(\mathsf{SDK}_R, \mathsf{VEK}_R)$. Let $pub = \{\mathsf{VEK}_S, \mathsf{VEK}_R\}$ be the public information. We set the signing key to $\mathsf{SK} = \{\mathsf{SDK}_S, pub\}$, and the verification key to $\mathsf{VK} = \{\mathsf{SDK}_R, pub\}$. Namely, the public verification key (available to the adversary) *contains the secret key of the receiver R.* To sign a message m, $\mathsf{Sig}(m)$ outputs $u = \mathsf{SigEnc}(m)$, while the verification algorithm $\mathsf{Ver}(u)$ runs $m \leftarrow \mathsf{VerDec}(u)$ and outputs succeed iff $m \neq \perp$. We note that the verification is indeed polynomial time since VK includes SDK_R.
- **Encryption $\mathcal{E}$.** The generation algorithm Enc-Gen runs $\mathsf{Gen}(1^k)$ twice to produce two key pairs $(\mathsf{SDK}_S, \mathsf{VEK}_S)$ and $(\mathsf{SDK}_R, \mathsf{VEK}_R)$. Let $pub = \{\mathsf{VEK}_S, \mathsf{VEK}_R\}$ be the public information. We set the encryption key to $\mathsf{EK} = \{\mathsf{SDK}_S, pub\}$, and the decryption key to $\mathsf{DK} = \{\mathsf{SDK}_R, pub\}$. Namely, the public encryption key (available to the adversary) *contains the secret key of the sender S.* To encrypt a message m, $\mathsf{Enc}(m)$ outputs $u = \mathsf{SigEnc}(m)$, while the decryption algorithm $\mathsf{Dec}(u)$ simply outputs $\mathsf{VerDec}(u)$. We note that the encryption is indeed polynomial time since EK includes SDK_S.

We say that the signcryption is *Insider-secure* against the corresponding attack (e.g. gCCA2/CMA) on the privacy/authenticity property, if the corresponding induced encryption/signature is secure against the same attack.[3] We will aim to satisfy IND-gCCA2-security for encryption, and UF-CMA-security for signatures.

3.3 Some Discussion

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 to the contents of m. Put differently, s is unforgeable and publicly verifiable. On the other hand, non-repudiation does not *automatically* follow from the definition of signcryption. Signcryption only allows the *receiver* to be convinced that m was sent by S, but does not necessarily enable a third party to verify this fact.

We believe that non-repudiation should not be part of the *definition* of signcryption security, but we will point out which of our schemes achieves it. Indeed, non-repudiation might be needed in some applications, while explicitly undesirable in others (e.g., this issue is the essence of undeniable [10] and chameleon [24] signature schemes).

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 SDK_R (since this induces a regular signature scheme), or may be possible by other means (like an appropriate zero-knowledge proof). In contrast, Outsider-security leaves open the possibility that R can generate — using its secret key — valid signcryptions of messages that were not actually sent by S. In such a case, non-repudiation cannot be achieved no matter what R does.

[3] One small technicality for the gCCA2-security. Recall, the equivalence relation $\mathcal{R}$ can depend on the public encryption key — in this case $\{\mathsf{SDK}_S, pub\}$. We strengthen this and allow it to depend only on pub (i.e. disallow the dependence on sender's secret key SDK_S).

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, the Outsider-security protects the privacy of R when talking to S from outside intruders, who do not know the secret key of S. On the other hand, Insider-security assumes that the sender S *is* the intruder attacking the privacy of R. But since S is the *only* party 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 R is the intruder attacking the authenticity of S, and simultaneously the *only* party that needs to be convinced of the authenticity of the (received) data. And, indeed, 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. We give just one example. Assume an adversary $\mathcal{A}$ happens to steal the key of S. Even though now $\mathcal{A}$ can send fake messages "from S to R", we still might not want $\mathcal{A}$ to understand previous (or even future) recorded signcryptions sent from honest S to R. Insider-security will guarantee this fact, 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 and 3. Other examples are given in [2], who show that authenticated encryption in the *symmetric setting* could be used to build Outsider-secure signcryption which is not Insider-secure. To summarize, one should carefully examine if one really needs the extra guarantees of Insider-security.

4 Two Sequential Compositions of Encryption and Signature

In this section, we will discuss two methods of constructing signcryption schemes that are based on sequential generic composition of encryption and signature: encrypt-then-sign ($\mathcal{E}t\mathcal{S}$) and sign-then-encrypt ($\mathcal{S}t\mathcal{E}$).

SYNTAX. Let $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme and $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a signature scheme. Both $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ have the same generation algorithm $\mathsf{Gen}(1^k)$. It runs $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}Gen}(1^k)$, $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$ and sets $\mathsf{VEK} = (\mathsf{VK}, \mathsf{EK})$, $\mathsf{SDK} = (\mathsf{SK}, \mathsf{DK})$. To describe the signcryptions from sender S to receiver R more compactly, we use the shorthands $\mathsf{Sig}_S(\cdot)$, $\mathsf{Enc}_R(\cdot)$, $\mathsf{Ver}_S(\cdot)$ and $\mathsf{Dec}_R(\cdot)$ indicating whose keys are used but omitting which specific keys are used, since the latter is obvious (indeed, Sig_S always uses SK_S, Enc_R — EK_R, Ver_S — VK_S and Dec_R — DK_R).

Now, we define "encrypt-then-sign" scheme $\mathcal{E}t\mathcal{S}$ by $u \leftarrow \mathsf{SigEnc}(m; (\mathsf{SK}_S, \mathsf{EK}_R)) = \mathsf{Sig}_S(\mathsf{Enc}_R(m))$. To de-signcrypt u, we let $\tilde{m} = \mathsf{Dec}_R(\mathsf{Msg}(u))$ provided $\mathsf{Ver}_S(u) = \mathsf{succeed}$, and $\tilde{m} = \perp$ otherwise. We then define $\mathsf{VerDec}(u; (\mathsf{DK}_R, \mathsf{VK}_S)) = \tilde{m}$. Notice, we do not mention $(\mathsf{EK}_S, \mathsf{DK}_S)$ and $(\mathsf{SK}_R, \mathsf{VK}_R)$, since they are not used to send the message from S to R. Similarly, we define "sign-then-encrypt" scheme $\mathcal{S}t\mathcal{E}$ by $u \leftarrow \mathsf{SigEnc}(m; (\mathsf{SK}_S, \mathsf{EK}_R)) = \mathsf{Enc}_R(\mathsf{Sig}_S(m))$. To de-signcrypt u, we let $s = \mathsf{Dec}_R(u)$, and set $\tilde{m} = \mathsf{Msg}(s)$ provided $\mathsf{Ver}_S(s) = \mathsf{succeed}$, and $\tilde{m} = \perp$ otherwise. We then define $\mathsf{VerDec}(u; (\mathsf{DK}_R, \mathsf{VK}_S)) = \tilde{m}$.

INSIDER-SECURITY. We now show that both $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ are secure composition paradigms. That is, they *preserve* (in terms of Insider-security) or even *improve* (in terms of Outsider-security) the security properties of $\mathcal{E}$ and $\mathcal{S}$. We start with Insider-security.

Theorem 1 *If $\mathcal{E}$ is* IND-gCCA2*-secure, and $\mathcal{S}$ is* UF-CMA*-secure, then $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ are both* IND-gCCA2*-secure and* UF-CMA*-secure in the Insider-security model.*

The simple proof of this result is given in Appendix A. However, we remark the crucial use of gCCA2-security when proving the security of $\mathcal{E}t\mathcal{S}$. Indeed, we can call two signcryptions u_1 and u_2 equivalent for

$\mathcal{E}t\mathcal{S}$, if each u_i is a valid signature (w.r.t. $\mathcal{S}$) of $e_i = \mathsf{Msg}(u_i)$, and e_1 and e_2 are equivalent (e.g., equal) w.r.t. to the equivalence relation of $\mathcal{E}$. In other words, a *different* signature of the *same* encryption clearly corresponds to the same message, and we should not reward the adversary for achieving such a trivial[4] task.

Remark 1 *We note that $\mathcal{S}t\mathcal{E}$ achieves non-repudiation. On the other hand, $\mathcal{E}t\mathcal{S}$ might not achieve obvious non-repudiation, except for some special cases. One such important case concerns encryption schemes, where the decryptor can reconstruct the randomness r used by the encryptor. In this case, presenting r such that $\mathsf{Enc}_R(m;\ r) = e$, and u is a valid signature of e yields non-repudiation.*

We note that, for the *Insider-security* in the public-key setting, we cannot hope to *amplify* the security of the "base" signature or encryption, unlike the symmetric setting, where a proper use of a MAC allows one to increase the privacy from CPA to CCA2-security (see [5, 23]). For example, in the Insider-security for encryption, the adversary is acting as the sender and holds the signing key. Thus, it is obvious that the use of this signing key cannot protect the receiver and increase the quality of the encryption. Similar argument holds for signatures. Thus, the result of Theorem 1 is the most optimistic we can hope for in that it at least *preserves* the security of the base signature and encryption, while simultaneously achieving *both* functionalities.

Outsider-security. On the other hand, 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 [5, 23, 1]. This shows that Outsider-security model is quite similar to the symmetric setting: namely, from the adversarial point of view the sender and the receiver "share" the secret key $(\mathsf{SDK}_S, \mathsf{SDK}_R)$.

Theorem 2 *If $\mathcal{E}$ is* IND-CPA*-secure, and $\mathcal{S}$ is* UF-CMA*-secure, then $\mathcal{E}t\mathcal{S}$ is* IND-gCCA2*-secure in the Outsider- and* UF-CMA*-secure in the Insider-security models.*

The formal proof is given in Appendix B. Intuitively, either the de-signcryption oracle always returns $\perp$ to the gCCA2-adversary, in which case it is "useless" and IND-CPA-security of $\mathcal{E}$ is enough, or the adversary can submit a valid signcryption $u = \mathsf{Sig}(\mathsf{Enc}(\cdot))$ to this oracle, in which case it breaks the UF-CMA-security of the "outside" signature $\mathcal{S}$.

Theorem 3 *If $\mathcal{E}$ is* IND-gCCA2*-secure, and $\mathcal{S}$ is* UF-NMA*-secure, then $\mathcal{S}t\mathcal{E}$ is* IND-gCCA2*-secure in the Insider- and* UF-CMA*-secure in the Outsider-security models.*

The formal proof is given in Appendix C. Intuitively, the IND-gCCA2-security of the "outside" encryption $\mathcal{E}$ makes the CMA attack of UF-CMA-adversary $\mathcal{A}$ "useless", by effectively hiding the signatures corresponding to $\mathcal{A}$'s queried messages, hence making the attack reduced to NMA.

5 Parallel Encrypt and Sign

So far we concentrated on two basic sequential composition methods, "encrypt-then-sign" and "sign-then-encrypt". Another natural generic composition method would be to both encrypt the message and sign the message, denoted $\mathcal{E}\&\mathcal{S}$. This operation simply outputs a pair (s, e), where $s \leftarrow \mathsf{Sig}_S(m)$ and $e \leftarrow \mathsf{Enc}_R(m)$. One should observe that $\mathcal{E}\&\mathcal{S}$ preserves the authenticity property but obviously does *not* preserve the privacy of the message as the signature s might reveal information about the message m.

[4]The task is indeed trivial in the Insider-security model, since the adversary has the signing key.

Moreover, if the adversary knows that $m \in \{m_0, m_1\}$ (as is the case for IND-security), it can see if s is a signature of m_0 or m_1, thus breaking IND-security. This simple observation was also made by [5, 23]. However, we would like to stress that this scheme has a great advantage: it allows one to parallelize the expensive public key operations, which could imply significant efficiency gains.

Thus, the question which arises is under which conditions can we design a secure signcryption scheme which would also yield itself to efficiency improvements such as parallelization of operations. More concretely, there is no reason why we should apply Enc_R and Sig_S to m itself. What if we apply some efficient "pre-processing" transformation T to the message m, which produces a pair (c, d), and then sign c and encrypt d in parallel? Under which conditions on T will this yield a secure signcryption? Somewhat surprisingly, we show a very general result: instantiating T as a commitment scheme would enable us to both achieve a signcryption scheme and parallelize the expensive public key operations. More precisely, *relaxed commitment is necessary and sufficient*! In the following we explain this result in more detail.

SYNTAX. Clearly, the values (c, d) produced by $T(m)$ should be such that m is recoverable from (c, d), But which exactly the *syntax* (but not yet the *security*) of a commitment scheme, as defined in Section 2.3. Namely, T could be viewed as the message commitment algorithm Commit, while the message recovery algorithm is the opening algorithm Open, and we want $\mathsf{Open}(\mathsf{Commit}(m)) = m$. For a technical reason, we will also assume there exists at most one valid c for every value of d. This is done without loss of generally when commitment schemes are used. Indeed, essentially all commitment schemes have, and can always be assumed to have, $d = (m, r)$, where r is the randomness of $\mathsf{Commit}(m)$, and $\mathsf{Open}(c, (m, r))$ just checks if $\mathsf{Commit}(m; r) = (c, (m, r))$ before outputting m.

Now, given any such (possibly insecure) $\mathcal{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, an encryption scheme $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a signature scheme $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$, we define a new composition paradigm, which we call "commit-then-encrypt-and-sign": shortly, $\mathcal{C}t\mathcal{E}\&\mathcal{S} = (\mathsf{Gen}, \mathsf{SigEnc}, \mathsf{VerDec})$. For simplicity, we assume for now that all the participants share the same common commitment key CK (e.g., generated by a trusted party). $\mathsf{Gen}(1^k)$ is the same as for $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ compositions: set $\mathsf{VEK} = (\mathsf{VK}, \mathsf{EK})$, $\mathsf{SDK} = (\mathsf{SK}, \mathsf{DK})$. Now, to signcrypt a message m from S to R, the sender S first runs $(c, d) \leftarrow \mathsf{Commit}(m)$, and outputs signcryption $u = (s, e)$, where $s \leftarrow \mathsf{Sig}_S(c)$ and $e \leftarrow \mathsf{Enc}_R(d)$. Namely, we sign the commitment c and encrypt the decommitment d. To de-signcrypt, the receiver R validates $c = \mathsf{Msg}(s)$ using $\mathsf{Ver}_S(s)$ and decrypts $d = \mathsf{Dec}_R(e)$ (outputting $\perp$ if either fails). The final output is $\tilde{m} = \mathsf{Open}(c, d)$. Obviously, $\tilde{m} = m$ if everybody is honest.

MAIN RESULT. We have defined the new composition paradigm $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ based purely on the syntactic properties of $\mathcal{C}$, $\mathcal{E}$ and $\mathcal{S}$. Now we formulate which security properties of $\mathcal{C}$ are necessary and sufficient so that our signcryption $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ preserves the security of $\mathcal{E}$ and $\mathcal{S}$. As in Section 4, we concentrate on UF-CMA and IND-gCCA2 security. Our main result is as follows:

Theorem 4 *Assume that $\mathcal{E}$ is* IND-gCCA2*-secure, $\mathcal{S}$ is* UF-CMA*-secure and $\mathcal{C}$ satisfies the syntactic properties of a commitment scheme. Then, in the Insider-security model, we have:*

- *$\mathcal{C}t\mathcal{E}\&\mathcal{S}$ is* IND-gCCA2*-secure $\Longleftrightarrow$ $\mathcal{C}$ satisfies the hiding property.*
- *$\mathcal{C}t\mathcal{E}\&\mathcal{S}$ is* UF-CMA*-secure $\Longleftrightarrow$ $\mathcal{C}$ satisfies the relaxed binding property.*

Thus, $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ preserves security of $\mathcal{E}$ and $\mathcal{S}$ iff $\mathcal{C}$ is a secure relaxed commitment. In particular, any secure regular commitment $\mathcal{C}$ yields secure signcryption $\mathcal{C}t\mathcal{E}\&\mathcal{S}$.

We prove our theorem by proving two related lemmas of independent interest. Define auxiliary encryption scheme $\mathcal{E}' = (\mathsf{Enc\text{-}Gen}', \mathsf{Enc}', \mathsf{Dec}')$ where (1) $\mathsf{Enc\text{-}Gen}' = \mathsf{Enc\text{-}Gen}$, (2) $\mathsf{Enc}'(m) = (c, \mathsf{Enc}(d))$, where $(c, d) \leftarrow \mathsf{Commit}(m)$, and (3) $\mathsf{Dec}'(c, e) = \mathsf{Open}(c, \mathsf{Dec}(d))$.

Lemma 1 *Assume $\mathcal{E}$ is* IND-gCCA2*-secure encryption. Then $\mathcal{E}'$ is* IND-gCCA2*-secure encryption iff $\mathcal{C}$ satisfies the hiding property.*

The proof is given in Appendix D. Intuitively, the hiding property is necessary since c is given "in the clear", and is sufficient since $\mathcal{E}$ is IND-gCCA2-secure and there is at most one valid value of c corresponding to every d (by our assumption).

We note that the first part of Theorem 4 follows using exactly the same proof as Lemma 1. Only few small changes (omitted) are needed due to the fact that the commitment is now signed. We remark only that IND-gCCA2 security is again important here. Informally, IND-gCCA2-security is robust to easily recognizable and invertible changes of the ciphertext. Thus, signing the commitment part — which is polynomially verifiable — does not spoil IND-gCCA2-security.

We now move to the second lemma. We define auxiliary signature scheme $\mathcal{S}' = (\mathsf{Sig\text{-}Gen}', \mathsf{Sig}', \mathsf{Ver}')$ as follows: (1) $\mathsf{Sig\text{-}Gen}' = \mathsf{Sig\text{-}Gen}$, (2) $\mathsf{Sig}'(m) = (\mathsf{Sig}(c), d)$, where, $(c, d) \leftarrow \mathsf{Commit}(m))$, (3) $\mathsf{Ver}'(s, d) = \mathsf{succeed}$ iff $\mathsf{Ver}(s) = \mathsf{succeed}$ and $\mathsf{Open}(\mathsf{Msg}(s), d) \neq \perp$.

Lemma 2 *Assume $\mathcal{S}$ is* UF-CMA*-secure signature. Then $\mathcal{S}'$ is* UF-CMA*-secure signature iff $\mathcal{C}$ satisfies the relaxed binding property.*

The proof is given in Appendix E. Intuitively, modulo breaking the security of the signature $\mathcal{S}$, the only way to forge a new signature in $\mathcal{S}'$ is to "reuse" some $\mathsf{Sig}(c)$ and come up with a some d' which opens c different from the original decommitment d. And this is exactly what the relaxed binding property of $\mathcal{C}$ guarantees. Notice, the binding can indeed be relaxed, since the value of c is chosen by the signing oracle and not by the adversary.

We note that the second part of Theorem 4 follows using exactly the same proof as Lemma 2. Only few small changes are needed due to the fact that the decommitment is now encrypted (e.g., the adversary chooses its own encryption keys and performs decryptions on its own). This completes the proof of Theorem 4.

Remark 2 *We note that $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ achieves non-repudiation by Lemma 2. Also note that the necessity of relaxed commitments holds in the weaker Outsider-security model as well. Finally, we note that $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm successfully applies to the symmetric setting as well.*

Remark 3 *We remark that in practice, $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ could be faster or slower than the sequential $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ compositions, depending on the specifics $\mathcal{C}$, $\mathcal{E}$ and $\mathcal{S}$. For most efficiency on the commitment side, however, one can use the simple commitment $c = H(m, r)$, $d = (m, r)$, where r is a short random string and H is a cryptographic hash function (analyzed as a random oracle). For provable security, one can use an almost equally efficient commitment scheme of [11, 19] based on CRHF's.*

6 On-line/Off-line Signcryption

Public-key operations are expensive. Therefore, we examine the possibility of designing signcryption schemes which could be run in two phases: (1) the *off-line* phase, performed before the messages to be signcrypted is known; and (2) the *on-line* phase, which uses the message and the pre-computation of the off-line stage, to efficiently produce the required signcryption. We show that the $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm is ideally suited for such a task, but first we recall a similar notion for ordinary signatures.

On-line/Off-line Signatures. On-line/Off-line signatures where introduced by Even et al. [15] who presented a general methodology to transform any signature scheme into a more efficient on-line/off-line signature (by using so called "one-time" signatures). Their construction, however, is mainly of theoretical interest. Recently, Shamir and Tauman [31] introduced the following much more efficient method to generate on-line/off-line signatures, which they called "hash-sign-switch". The idea is to use *trapdoor commitments* (see Section 2.3) in the following way. The signer S chooses two pairs of keys: regular signing keys $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$, and trapdoor commitment keys $(\mathsf{TK}, \mathsf{CK}) \leftarrow \mathsf{Setup}(1^k)$. S keeps $(\mathsf{SK}, \mathsf{TK})$ secret, and publishes $(\mathsf{VK}, \mathsf{CK})$. In the off-line phase, S prepares $(c, d_0) \leftarrow \mathsf{Commit}_{\mathsf{CK}}(0)$, and $s \leftarrow \mathsf{Sig}_{\mathsf{SK}}(c)$. In the on-line phase, when the message m arrives, S creates "fake" decommitment $(c, d) \leftarrow \mathsf{Switch}_{\mathsf{TK}}((c, d_0), m)$ to m, and outputs (s, d) as the signature. To verify, the receiver R checks that s is a valid signature of $c = \mathsf{Msg}(s)$, and $\mathsf{Open}_{\mathsf{CK}}(c, d) = m$.

Notice, this is very similar to the auxiliary signature scheme $\mathcal{S}'$ we used in Lemma 2. The only difference is that the "fake" pair (c, d) is used instead of $\mathsf{Commit}(m)$. However, by the trapdoor collisions property of trapdoor commitments, we get that $(c, d) \approx \mathsf{Commit}(m)$, and hence Lemma 2 — true for any commitment scheme — implies that this modified signature scheme is indeed secure (more detailed proof is given in [31]). Thus, the resulting signature $\mathcal{S}''$ essentially returns the same $(\mathsf{Sig}(c), d)$ as $\mathcal{S}'$, except that the expensive signature Sig is computed in the off-line phase.

"Hash-Sign-Switch" for Signcryption. Now, we could use the on-line/off-line signature $\mathcal{S}''$ above with any of our composition paradigms: $\mathcal{E}t\mathcal{S}, \mathcal{S}t\mathcal{E}$ or $\mathcal{C}t\mathcal{E}\&\mathcal{S}$. In all cases this would move the actual signing operation into the off-line phase. For example, $\mathcal{E}t\mathcal{S}$ will (essentially) return $(\mathsf{Sig}(c(e)), d(e))$, where $e \leftarrow \mathsf{Enc}(m)$; while $\mathcal{S}t\mathcal{E}$ will return $\mathsf{Enc}(\mathsf{Sig}(c(m)), d(m))$. We could also apply it "directly" to the $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ scheme. However, $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ scheme *already uses commitments*! So let us see what happens when we use a trapdoor commitment $\mathcal{C}$ instead of any general commitment. We see that we still return $(\mathsf{Sig}(c), \mathsf{Enc}(d))$ (where $(c, d) \leftarrow \mathsf{Switch}(\mathsf{Commit}(0), m) \approx \mathsf{Commit}(m)$), except the expensive signature part is performed off-line, exactly as we wish. Thus, $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ yields a more efficient (and provably secure by Theorem 4) on-line/off-line implementation than the one we get by blindly applying the "hash-sign-switch" technique to the $\mathcal{E}t\mathcal{S}$ or $\mathcal{S}t\mathcal{E}$ schemes.

We remark that in this scheme the trapdoor key TK has to be known to the sender, but not to the receiver. Hence, each user P has to generate its own pair $(\mathsf{TK}, \mathsf{CK})$ during key generation, keeping TK as part of SDK_P. Also, P should use its own CK_P when sending messages, and the sender's CK when receiving messages. Notice, since trapdoor commitments are *information-theoretically* hiding, there is no danger for the receiver that the sender chooses a "bad" commitment key (the hiding property is satisfied for all CK's, and it is in sender's interest to choose CK so that the binding is satisfied as well).

Adding On-line/Off-line Encryption. We have successfully moved the expensive public-key signature to the off-line phase. What about public-key encryption? We can use the folklore technique of integrating public- and secret-key encryptions: $\mathsf{Enc}'_{\mathsf{EK}}(m) = (\mathsf{Enc}_{\mathsf{EK}}(r), E_r(m))$. Namely, we encrypt a random secret-key r for symmetric encryption E, and then encrypt the actual message m using E with the key r. Clearly, we can do the (much more expensive) public-key encryption $\mathsf{Enc}_{\mathsf{EK}}(r)$ in the off-line stage. Surprisingly, this folklore technique, which is being extensively used in practice, has only recently been formally analyzed in the CCA2-setting by [12]. Translated to our terminology, IND-gCCA2-secure Enc and E yield IND-gCCA2-secure Enc' above ([12] showed this for regular IND-CCA2-security). As a side remark, in the random oracle model, clever integration of public- and secret-key encryption allows us to get IND-CCA2-secure Enc' starting from much less secure base encryption Enc (e.g., see [16, 26]). Thus, making encryption off-line can also amplify its security in this setting.

Final Scheme. To summarize, we get the following very efficient on-line/off-line signcryption scheme from any signature $\mathcal{S}$, public-key encryption $\mathcal{E}$, trapdoor commitment $\mathcal{C}$, and symmetric encryption

E: (1) in the off-line stage generate $(c, d_0) \leftarrow \mathsf{Commit}_{\mathsf{CK}_S}(0)$, and prepare $e_1 \leftarrow \mathsf{Enc}_{\mathsf{EK}_R}(r)$, and $s \leftarrow \mathsf{Sig}_{\mathsf{SK}_S}(c)$; (2) in the on-line stage, create $(c, d) \leftarrow \mathsf{Switch}_{\mathsf{TK}_S}((c, d_0), m)$, $e_2 \leftarrow E_r(d)$, and return $(s, (e_1, e_2))$. In essence, we efficiently compute and return $(\mathsf{Sig}(c), (\mathsf{Enc}(r), E_r(d)))$, where $(c, d) \approx \mathsf{Commit}(m)$. Since the switching operation and the symmetric encryption are usually very fast, we get significant efficiency gain. Decryption and verification are obvious.

7 Multi-User Setting

Even though it is easier to work in the two-user setting, we argued in detail in Section 3.1 that one really needs multi-user security for most applications of signcryption. This is precisely the topic of this section.

SYNTAX. So far we have concentrated on the 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 now have identities. We denote by ID_P the identity of user P. 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 VEK_P from ID_P (e.g., ID_P could be VEK_P). Next, we change the syntax of the signcryption algorithm SigEnc to both take and output the identity of the sender and the receiver. Specifically, (1) the signcryption for user S, on input, $(m, \mathsf{ID}_{S'}, \mathsf{ID}_{R'})$, uses $\mathsf{VEK}_{R'}$ and generates $(u, \mathsf{ID}_S, \mathsf{ID}_{R'})$ provided $\mathsf{ID}_S = \mathsf{ID}_{S'}$; (2) the de-signcryption for user R, on input $(u, \mathsf{ID}_{S'}, \mathsf{ID}_{R'})$, uses $\mathsf{VEK}_{S'}$ and outputs $\tilde{m}$ provided $\mathsf{ID}_R = \mathsf{ID}_{R'}$. It must be clear from which S' the message $\tilde{m}$ came from. Otherwise this will not be able to satisfy the security property described below.

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 beside SDK_S and SDK_R, and has access to the signcryption oracle of S (which it can call with *any* $\mathsf{ID}_{R'}$ and not just ID_R) and the de-signcryption oracle for R (which it can call with *any* $\mathsf{ID}_{S'}$ and not just ID_S). Naturally, to break the UF-CMA-security, $\mathcal{A}$ has to come up with a valid signcryption $(u, \mathsf{ID}_S, \mathsf{ID}_R)$ of the message m such that $(m, \mathsf{ID}_S, \mathsf{ID}_R)$ was not queried earlier to the signcryption oracle of S. Similarly, to break IND-gCCA2-security of encryption, $\mathcal{A}$ has to come up with m_0 and m_1 such that it can distinguish $\mathsf{SigEnc}(m_0, \mathsf{ID}_S, \mathsf{ID}_R)$ from $\mathsf{SigEnc}(m_1, \mathsf{ID}_S, \mathsf{ID}_R)$. Of course, given a challenge $(u, \mathsf{ID}_S, \mathsf{ID}_R)$, $\mathcal{A}$ is disallowed to ask the de-signcryption oracle for R a query $(u', \mathsf{ID}_S, \mathsf{ID}_R)$ where $\mathcal{R}(u, u') = true$.

We define Insider-security in an analogous manner, except now the adversary has all the secret keys except SDK_S when attacking authenticity or SDK_R when attacking privacy. Also, for UF-CMA-security, a forgery $(u, \mathsf{ID}_S, \mathsf{ID}_{R'})$ of a message m is "new" as long as $(m, \mathsf{ID}_S, \mathsf{ID}_{R'})$ was not queried (even though $(m, \mathsf{ID}_S, \mathsf{ID}_{R''})$ could be queried). Similarly, $\mathcal{A}$ could choose to distinguish signcryptions $(m_0, \mathsf{ID}_{S'}, \mathsf{ID}_R)$ from $(m_1, \mathsf{ID}_{S'}, \mathsf{ID}_R)$ (for any S'), and only has the natural restriction on asking de-signcryption queries of the form $(u, \mathsf{ID}_{S'}, \mathsf{ID}_R)$, but has no restrictions on using $\mathsf{ID}_{S''} \neq \mathsf{ID}_{S'}$.

EXTENDING SIGNCRYPTION. We can see that the signcryption algorithms that we use so far have to be upgraded, so that they use the new inputs ID_S and ID_R in non-trivial manner. For example, if the $\mathcal{E}t\mathcal{S}$ method is used in the multi-user setting, the adversary $\mathcal{A}$ can easily break the gCCA2-security, even in the Outsider-model. Indeed, given the challenge $u = (\mathsf{Sig}_S(e), \mathsf{ID}_S, \mathsf{ID}_R)$, where $e = \mathsf{Enc}_R(m_b)$, $\mathcal{A}$ can replace the sender's signature with its own by computing $u' = (\mathsf{Sig}_{\mathcal{A}}(e), \mathsf{ID}_{\mathcal{A}}, \mathsf{ID}_R)$ and ask R to de-signcrypt it. Since $\mathcal{A}$ has no restrictions on using $\mathsf{ID}_{\mathcal{A}} \neq \mathsf{ID}_S$ in its de-signcryption oracle queries, $\mathcal{A}$ can effectively obtain the decryption of e (i.e. m_b). Similar attack on encryption holds for the $\mathcal{S}t\mathcal{E}$ scheme, while in $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ both the encryption and the signature suffer from these trivial attacks. In other words, the schemes we designed so far suffer from the identity fraud, since the signcryption does not "bind together" the message with the identities of the sender and the recipient, allowing the adversary to freely tamper with these identities.

It turns out there is a general simple solution to this problem for all the schemes we considered so far: namely, the schemes build from general signature and encryption. Specifically, whatever security was proven in the two-user setting remains unchanged for the multi-user setting as long as we follow these simple changes to "bind" the message and the identities:

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 either the identity of the sender or of the receiver do not match what is expected, output $\perp$.

Hence, we get the following new analogs for $\mathcal{E}t\mathcal{S}$, $\mathcal{S}t\mathcal{E}$ and $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ in the multi-user setting:

- $\mathcal{E}t\mathcal{S}$ returns $(\mathsf{Sig}_S(\mathsf{Enc}_R(m, \mathsf{ID}_S), \mathsf{ID}_R), \mathsf{ID}_S, \mathsf{ID}_R)$.
- $\mathcal{S}t\mathcal{E}$ returns $(\mathsf{Enc}_R(\mathsf{Sig}_S(m, \mathsf{ID}_R), \mathsf{ID}_S), \mathsf{ID}_S, \mathsf{ID}_R)$.
- $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ returns $(\mathsf{Sig}_S(c, \mathsf{ID}_R), \mathsf{Enc}_R(d, \mathsf{ID}_S), \mathsf{ID}_S, \mathsf{ID}_R)$, where $(c, d) \leftarrow \mathsf{Commit}(m)$.

The formal justification for our generic transformation follows by a tedious, but straightforward extension of the corresponding proofs for two-user setting. We omit the details to the full version.

8 On CCA2 Security and Strong Unforgeability

This section will be mainly dedicated to the conventional notion of CCA2-attack for encryption. Much of the discussion also applies to a related notion of strong unforgeability, sUF, for signatures. Despite the fact that one specifies the attack model, and the other — the adversary's goal, we will see that the relation between gCCA2/CCA2, and UF/sUF notions is quite similar. We will argue that: (1) gCCA2-attack and UF-security are better suited for a "good" *definition* than their stronger but syntactically ill CCA2 and sUF counterparts; (2) it is unlikely that the extra strength of CCA2 w.r.t. gCCA2 and sUF w.r.t. UF will find any useful applications.

Of course, what is stated above is a subjective opinion. Therefore, for complete reference we briefly indicate in Appendix F which of our previous results for signcryption (stated for gCCA2/UF notions) extend to the CCA2/sUF notions. Roughly, half of the implications still hold, while the other half fails to do so. As one representative example, $\mathcal{E}t\mathcal{S}$ is no longer CCA2-secure even if $\mathcal{E}$ is CCA2-secure. A "counter-example" comes when we use a perfectly possible UF-CMA-secure signature scheme $\mathcal{S}$ which always appends a useless bit during signing. By simply flipping this bit on the challenge ciphertext, CCA2-adversary is now "allowed" to use the decryption oracle and recover the plaintext. The artificial nature of this "counter-example" is perfectly highlighted by Theorem 1, which shows that the IND-gCCA2-security of $\mathcal{E}t\mathcal{S}$ is preserved.

Definitional Necessity. Even more explicitly, appending a useless (but harmless) bit to a CCA2-secure encryption no longer leaves it CCA2-secure. It seems a little disturbing that this clearly harmless (albeit useless) modification does not satisfy the *definition* of "secure encryption". The common answer to the above criticism is that there is nothing wrong if we became overly strict with our definitions, as long as (1) the definitions do not allow for "insecure" schemes; and (2) we can meet them. In other words, the fact that some secure, but "useless" constructions are ruled out can be tolerated. However, as we illustrated for the first time, the conventional CCA2 notion *does* rule out some secure "useful" constructions as well. For example, it might have led one to believe that the $\mathcal{E}t\mathcal{S}$ scheme is generically insecure and should be avoided, while we showed that this is not the case.

Relation to Non-malleability. We recall that the concept of indistinguishability is very useful in terms of *proving* schemes secure, but it is not really "natural". It is generally believed that a more useful security notion — and the one really important in applications — is that of *non-malleability* [14] (denoted NM), which we explain in a second. Luckily, it is known [14, 4] that IND-CCA2 is equivalent to NM-CCA2, which "justifies" the use of IND-CCA2 as a simpler notion to work with. And now that we relaxed IND-CCA2 to IND-gCCA2, a valid concern arises that we loose the above equivalence, and therefore the justification for using indistinguishability as our security notion. A closer look, however, reveals that this concern is merely a syntactic triviality. Let us explain.

In essence, NM-security roughly states the following: upon seeing some unknown ciphertext e, the only thing the adversary can extract — which bears any relevance to the corresponding plaintext m — is the encryption of this plaintext (which the adversary has anyway). The current formalization of non-malleability additionally requires that the only such encryption e' that $\mathcal{A}$ can get is e itself. However, unlike the first property, the last requirement does not seem crucial, provided that *anybody can tell that the ciphertext e' encrypts the same message as e, by only looking at e and e'*. In other words, there could possibly be no harm even if $\mathcal{A}$ can generate $e' \neq e$: anyone can tell that $\mathsf{Dec}(e) = \mathsf{Dec}(e')$, so there is no point to even change e to e'. Indeed, we can relax the formalization of non-malleability (call if gNM) by using a decryption-respecting relation $\mathcal{R}$, just like we did for the CCA2 attack: namely, $\mathcal{A}$ is not considered successful if it outputs e' s.t. $\mathcal{R}(e, e') = \mathit{true}$. Once this is done, the equivalence between "gNM-CCA2" and IND-gCCA2 holds again.

Applicational Necessity. The above argument also indicates that gCCA2-security is sufficient for all applications where chosen ciphertext security matters (e.g., those in [32, 9, 8]). Moreover, it is probably still a slight overkill in terms of a necessary and sufficient formalization of "secure encryption" from the applicational point of view. Indeed, we tried to relax the notion of CCA2-security to the minimum extent possible, just to avoid the syntactic problems of CCA2-security. In particular, we are not aware of any "natural" encryption scheme in the gap between gCCA2 and CCA2-security.[5] The only thing we are saying is that the *notion* of gCCA2 security is more robust to syntactic issues, seems more applicable for studying generic properties of "secure encryption", while also being sufficient for its applications.

Strong Unforgeability. Finally, we briefly remark on the concept of sUF-security for signatures. To the best of our knowledge, the extra guarantees of this concept have no realistic applications (while suffering similar syntactic problems as CCA2-security does). Indeed, once the message m is signed, there is no use to produce a different signature of the same message: the adversary already has a valid signature of m. The only "application" we are aware of is building CCA2-secure encryption from a CPA-secure encryption, via the $\mathcal{E}t\mathcal{S}$ method. As we demonstrated in Theorem 2, sUF-security is no longer necessarily once we accept the concept of gCCA2-security.

[5] In other words, the separating examples we know about all start with a CCA2-secure encryption, and make a more complicated "counter-example" out of it. Very recently, a natural separation was given by [13], but for the setting of *broadcast* (rather than ordinary) encryption.

References

[1] J. An and M. Bellare, "Does encryption with redundancy provide authenticity?," In *Eurocrypt '01*, pp. 512–528, LNCS Vol. 2045.

[2] J. An and Y. Dodis, "Secure integration of symmetric- and public-key authenticated encryption." Manuscript, 2002.

[3] J. Baek, R. Steinfeld, and Y. Zheng, "Formal proofs for the security of signcryption," In *PKC '02*, 2002.

[4] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, "Relations among notions of security for public-key encryption schemes," In *Crypto '98*, LNCS Vol. 1462.

[5] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," In *Asiacrypt '00*, LNCS Vol. 1976.

[6] M. Bellare, P. Rogaway, "Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography," In *Asiacrypt '00*, LNCS Vol 1976.

[7] G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," *JCSS*, 37(2):156–189, 1988.

[8] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," In *Proc. 42st FOCS*, pp. 136–145. IEEE, 2001.

[9] R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," In *Eurocrypt '01*, pp. 453–474, LNCS Vol. 2045.

[10] D. Chaum and H. Van Antwerpen, "Undeniable signatures," In *Crypto '89*, pp. 212–217, LNCS Vol. 435.

[11] I. Damgård, T. Pedersen, and B. Pfitzmann, "On the existence of statistically hiding bit commitment schemes and fail-stop signatures," In *Crypto '93*, LNCS Vol. 773.

[12] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith, "Efficient and Non-interactive Non-malleable Commitment," In *Eurocrypt '01*, pp. 40–59, LNCS Vol. 2045.

[13] Y. Dodis, N. Fazio, "On Adaptive and CCA Security for Public Key Trace and Revoke Schemes", *manuscrypt*, 2002.

[14] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography," In *Proc. 23rd STOC*, ACM, 1991.

[15] S. Even, O. Goldreich, and S. Micali, "On-Line/Off-Line Digital Schemes," In *Crypto '89*, pp. 263–275, LNCS Vol. 435.

[16] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," In *Crypto '99*, pp. 537–554, 1999, LNCS Vol. 1666.

[17] S. Goldwasser and S. Micali, "Probabilistic encryption," *JCSS*, 28(2):270–299, April 1984.

[18] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Computing*, 17(2):281–308, April 1988.

[19] S. Halevi and S. Micali, "Practical and provably-secure commitment schemes from collision-free hashing," In *Crypto '96*, pp. 201–215, 1996, LNCS Vol. 1109.

[20] W. He and T. Wu, "Cryptanalysis and Improvement of Petersen-Michels Signcryption Schemes," *IEE Computers and Digital Communications*, 146(2):123–124, 1999.

[21] C. Jutla, "Encryption modes with almost free message integrity," In *Eurocrypt '01*, pp. 529–544, LNCS Vol. 2045.

[22] J. Katz and M. Yung, "Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation," In *FSE '00*, pp. 284–299, LNCS Vol. 1978.

[23] H. Krawczyk, "The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)," In *Crypto '01*, pp. 310–331, LNCS Vol. 2139.

[24] H. Krawczyk and T. Rabin, "Chameleon Signatures," In *NDSS '00*, pp. 143–154, 2000.

[25] M. Naor and M. Yung, "Universal One-Way Hash Functions and their Cryptographic Applications," In *Proc. 21st STOC*, pp. 33–43, ACM, 1989.

[26] T. Okamoto and D. Pointcheval, "React: Rapid enhanced-security asymmetric cryptosystem transform," In *CT-RSA '01*, pp. 159–175, 2001, LNCS Vol. 2020.

[27] H. Petersen and M. Michels, "Cryptanalysis and Improvement of Signcryption Schemes," *IEE Computers and Digital Communications*, 145(2):149–151, 1998.

[28] C. Rackoff and D. Simon, "Non-Interactive zero-knowledge proof of knowledge and chosen ciphertext attack," In *Crypto '91*, LNCS Vol. 576.

[29] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," In *Proc. 8th CCS*, ACM, 2001.

[30] C. Schnorr and M. Jakobsson, "Security of Signed ElGamal Encryption," In *Asiacrypt '00*, pp. 73–89, LNCS Vol. 1976.

[31] A. Shamir and Y. Tauman, "Improved Online/Offline Signature Schemes," In *Crypto '01*, pp. 355–367, LNCS Vol. 2139.

[32] V. Shoup, "On Formal Models for Secure Key Exchange," Technical Report RZ 3120, IBM Research, 1999.

[33] V. Shoup, "A proposal for an ISO standard for public key encryption (version 2.1)," Manuscript, Dec. 20, 2001.

[34] D. Simon, "Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?," In *Eurocrypt '98*, pp. 334–345, LNCS Vol. 1403.

[35] Y. Tsiounis and M. Yung, "On the Security of ElGamal Based Encryption," In *PKC '98*, pp. 117–134, LNCS Vol. 1431.

[36] Y. Zheng, "Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption)," In *Crypto '97*, pp. 165–179, 1997, LNCS Vol. 1294.

[37] Y. Zheng and H. Imai, "Efficient Signcryption Schemes on Elliptic Curves," *Information Processing Letters*, 68(5):227–233, December 1998.

A Proof of Theorem 1

We sketch the proofs of all four claims.

(1) UF-CMA-SECURITY OF $\mathcal{E}t\mathcal{S}$. Take any forger $\mathcal{A}'$ for the $\mathcal{E}t\mathcal{S}$. We can easily construct the forger $\mathcal{A}$ for $\mathcal{S}$ that has identical probability of forging signatures. $\mathcal{A}$ views the keys of $\mathcal{S}$ as $(\mathsf{SK}_S, \mathsf{VK}_S)$, and by itself picks a pair of encryption keys $(\mathsf{EK}_R, \mathsf{DK}_R) \leftarrow \mathsf{Enc\text{-}Gen}(1^k)$. $\mathcal{A}$ then hands $(\mathsf{EK}_R, \mathsf{DK}_R, \mathsf{VK}_S)$ to $\mathcal{A}'$, as the public key of the induced signature scheme. Next, $\mathcal{A}$ can easily simulate the signcryption query of $\mathcal{A}'$ for message m' by first creating $e' \leftarrow \mathsf{Enc}_R(m')$ and then asking the signing oracle for $\mathcal{S}$ to sign e'. Finally, when $\mathcal{A}'$ produces a forgery u for $\mathcal{E}t\mathcal{S}$, $\mathcal{A}$ outputs u as well. Notice that if u is a valid signcryption of some new message m (w.r.t. $\mathcal{E}t\mathcal{S}$), then u is a valid signature of $e = \mathsf{Msg}(u)$ (w.r.t. $\mathcal{S}$). Moreover, since m is new, $m = \mathsf{Dec}_R(e)$ and the encryption $\mathcal{E}$ is committing, e could not have been signed by the signing oracle, so e is indeed a new message w.r.t. $\mathcal{S}$.

(2) UF-CMA-SECURITY OF $\mathcal{S}t\mathcal{E}$. The proof is completely identical to the above with few minor differences. Now, $\mathcal{A}$ simulates the signcryption query m' of $\mathcal{A}'$ by first asking the oracle for $\mathcal{S}$ to produce the signature s' for m', and then returning $u' \leftarrow \mathsf{Enc}_R(s')$. Similarly, when $\mathcal{A}'$ produces a forged signcryption u of a new message m (w.r.t. $\mathcal{S}t\mathcal{E}$), $\mathcal{A}$ outputs $\mathsf{Dec}_R(u)$ (which it can do efficiently). Finally, notice that if u is a valid signcryption of some new m (w.r.t. $\mathcal{S}t\mathcal{E}$), then $s = \mathsf{Dec}_R(u)$ is a valid signature of this new m (w.r.t. $\mathcal{S}$).

(3) IND-gCCA2-SECURITY OF $\mathcal{E}t\mathcal{S}$. As a new element with gCCA2-security, we first have to deal with the equivalence relation. So let $\mathcal{R}$ be the equivalence relation w.r.t. which $\mathcal{E}$ is secure. We define the equivalence relation for the induced encryption $\mathcal{R}'$ for $\mathcal{E}t\mathcal{S}$ to be $\mathcal{R}'(u_1, u_2) = \mathcal{R}(e_1, e_2)$, if each u_i is valid signature of $e_i = \mathsf{Msg}(u_i)$ w.r.t. VK_R, and $false$ otherwise. We check that $\mathcal{R}'$ is decryption-respecting. Indeed, for $\mathcal{R}'(u_1, u_2) = true$, we have that each u_i is a valid signature of e_i, and also that $\mathsf{Dec}_R(e_1) = \mathsf{Dec}_R(e_2)$ (since $\mathcal{R}$ is decryption-respecting), which means that $\mathsf{VerDec}(u_1) = \mathsf{VerDec}(u_2)$ indeed.

Now, assume $\mathcal{E}t\mathcal{S}$ is not CCA2 secure w.r.t. $\mathcal{R}'$. We show that the same holds for $\mathcal{E}$ w.r.t. to $\mathcal{R}$ as well. To do this, we take any distinguisher $\mathcal{A}'$ for $\mathcal{E}t\mathcal{S}$ and construct $\mathcal{A}$ for $\mathcal{E}$ as follows. $\mathcal{A}$ views the keys of $\mathcal{E}$ as $(\mathsf{EK}_R, \mathsf{DK}_R)$, and by itself picks a pair of signing keys $(\mathsf{SK}_S, \mathsf{VK}_S) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$. $\mathcal{A}$ then hands $(\mathsf{EK}_R, \mathsf{SK}_S, \mathsf{VK}_S)$ to $\mathcal{A}'$, as the public key of the induced encryption scheme. To simulate the de-signcryption query u' made by $\mathcal{A}'$, $\mathcal{A}$ first checks that u' is a valid signature of $e' = \mathsf{Msg}(u')$ (if not, it returns $\perp$), and then asks its own decryption oracle to decrypt e'. Next, when $\mathcal{A}'$ outputs a pair of messages m_0 and m_1, $\mathcal{A}'$ outputs the same pair. Then, when $\mathcal{A}$ by itself gets the challenge $e = \mathsf{Enc}_R(m_b)$ (for unknown b), it hands $u \leftarrow \mathsf{Sig}_S(e)$ to $\mathcal{A}'$. Now, the definition of $\mathcal{R}'$ tells us that $\mathcal{A}'$ is disallowed to de-signcrypt any u' which a valid signature of e' satisfying $\mathcal{R}(e, e') = true$. But such e' are the only queries that $\mathcal{A}$ itself is disallowed to ask its decryption oracle! Thus, $\mathcal{A}$ can still handle all the legal de-signcryption queries of $\mathcal{A}'$, in the same manner as before. Finally, $\mathcal{A}$ outputs the same guess b' that $\mathcal{A}'$ outputs, which clearly gives $\mathcal{A}$ the same probability of being correct as $\mathcal{A}'$ has.

(4) IND-gCCA2-SECURITY OF $\mathcal{S}t\mathcal{E}$. Again, the proof is very similar to the above with few small differences. First, the relation $\mathcal{R}'$ is now the same as the relation $\mathcal{R}$. Notice, it is decryption-respecting w.r.t. $\mathcal{S}t\mathcal{E}$ since if $\mathcal{R}(u_1, u_2) = true$, it means that $s_1 = \mathsf{Dec}_R(u_1) = \mathsf{Dec}_R(u_2) = s_2$ (since $\mathcal{R}$ is decryption-respecting w.r.t. $\mathcal{E}$), and this implies that $m_1 = \mathsf{Msg}(s_1) = \mathsf{Msg}(s_2) = m_2$. Second, $\mathcal{A}$ simulates the de-signcryption query u' made by $\mathcal{A}'$, by first decrypting u' into s' (using its own decryption oracle), and then checking if s' is a valid signature of $m' = \mathsf{Msg}(s')$. Third, when $\mathcal{A}'$ outputs a pair of messages m_0 and m_1, $\mathcal{A}'$ outputs s_0 and s_1, where $s_i = \mathsf{Sig}_S(m_i)$ (notice, $\mathcal{A}$ can sign by itself), and then gives $\mathcal{A}'$ the same challenge $u = \mathsf{Enc}_R(s_b)$ it gets. Finally, the definition of $\mathcal{R}' = \mathcal{R}$ tells us

that $\mathcal{A}'$ is disallowed to de-signcrypt any u' satisfying $\mathcal{R}(u, u') = true$, which again are the only queries that $\mathcal{A}$ itself is disallowed to ask its decryption oracle!

B Proof of Theorem 2

The proof of UF-CMA-security is the same as in Theorem 1, since we did not use the security of encryption there. Thus, we concentrate on showing IND-gCCA2-security. First, we define the equivalence relation $\mathcal{R}'$ by: $\mathcal{R}'(u_1, u_2) = (e_1 \stackrel{?}{=} e_2)$, if each u_i is a valid signature of $e_i = \mathsf{Msg}(u_i)$ w.r.t. VK_R, and $false$ otherwise. We already checked in Theorem 1 that this $\mathcal{R}'$ is decryption-respecting (as it corresponds to $\mathcal{R}$ being the equality relation).

Next, let $\mathcal{A}'$ be the adversary breaking IND-gCCA2-security of $\mathcal{E}t\mathcal{S}$ in the Outsider-security model. Recall, $\mathcal{A}'$ only knows $pub = (\mathsf{EK}_R, \mathsf{VK}_S)$, but has access to the signcryption and the de-signcryption oracles SigEnc and VerDec. By assumption, $\Pr(\tilde{b} = b) \geq \frac{1}{2} + \varepsilon$, where the probability is taken over the all the randomness needed to perform the run of $\mathcal{A}'$ (as described in Section 3), b is the real index of the message being signcrypted, and $\tilde{b}$ is the guess of $\mathcal{A}'$. Let Forged denote the following event: somewhere during its run, $\mathcal{A}'$ called its de-signcryption oracle VerDec on some value u satisfying:

1. u is a valid signature of $e = \mathsf{Msg}(u)$ w.r.t. $\mathcal{S}$, i.e. $\mathsf{Ver}_S(u) = \mathsf{succeed}$; and
2. u was not given to $\mathcal{A}'$ by the signcryption oracle SigEnc so far.

Intuitively, if Forged happened, than $\mathcal{A}'$ broke the security of signature $\mathcal{S}$, else the de-signcryption oracle VerDec is useless, and $\mathcal{A}'$ breaks IND-CPA-security of $\mathcal{E}$. Formally, we have

$$\begin{aligned}
\varepsilon &\leq \Pr\left(b = \tilde{b}\right) - \frac{1}{2} = \left(\Pr(b = \tilde{b} \wedge \mathsf{Forged}) + \Pr(b = \tilde{b} \wedge \overline{\mathsf{Forged}})\right) - \frac{1}{2} \\
&\leq \Pr(\mathsf{Forged}) + \left(\Pr(b = \tilde{b} \wedge \overline{\mathsf{Forged}}) - \frac{1}{2}\right)
\end{aligned}$$

Hence, we either have $\Pr(\mathsf{Forged}) \geq \frac{\varepsilon}{2}$, or $\Pr(b = \tilde{b} \wedge \overline{\mathsf{Forged}}) \geq \frac{1}{2} + \frac{\varepsilon}{2}$. We treat these separately.

Case 1: $\Pr(\mathsf{Forged}) \geq \frac{\varepsilon}{2}$. We show that we can construct forger $\mathcal{A}_1$ which breaks UF-CMA-security of $\mathcal{S}$, which is a contradiction. Our forger $\mathcal{A}_1$ behaves exactly in the same way as the forger in the proof of Theorem 1, except: (1) it simulates the signcryption oracle SigEnc as before by asking its own signing oracle to sign $e \leftarrow \mathsf{Enc}_R(m)$; (2) it simulates the de-signcryption oracle (which was not necessary in Theorem 1) explicitly using its knowledge of DK_R which it had picked; (3) it does not wait for $\mathcal{A}'$ to complete, but instead waits for the event Forged to happen (which it can easily check by itself), and outputs forgery u when u triggering Forged is given by $\mathcal{A}'$; (4) it announces failure if Forged does not happen during the run of $\mathcal{A}'$. By the assumption that $\Pr(\mathsf{Forged}) \geq \varepsilon/2$, $\mathcal{A}_1$ indeed contradicts UF-CMA-security of $\mathcal{S}$.

Case 2: $\Pr(b = \tilde{b} \wedge \overline{\mathsf{Forged}}) \geq \frac{1}{2} + \frac{\varepsilon}{2}$. First, we note that when Forged does not happen, *all* the queries u' that $\mathcal{A}'$ gives to VerDec have one of the following two forms: (a) $\mathsf{Ver}_S(u') = \mathsf{fail}$; or (b) u' was already returned by SigEnc on some query m'. Well, in type (a) queries $\mathcal{A}'$ can check by itself that VerDec will return $\perp$ since the "outside" signature does not check, while in type (b) queries there was no need to submit u' in the first place, since $\mathcal{A}'$ already knows the answer m'. Overall, the de-signcryption oracle is useless: $\mathcal{A}'$ can compute all the answers by itself and CPA-security suffices.

Formally, we show that we can construct an adversary $\mathcal{A}_2$ which would break the IND-CPA-security of $\mathcal{E}$. Again, $\mathcal{A}_2$ behaves the same as the adversary used in Theorem 1, except: (1) it simulates the signcryption oracle (which was not necessary in Theorem 1) explicitly using its knowledge of SK_S which it had picked; (2) it keeps track of all the pairs (m', u') that were simulated by the signcryption oracle; (3) it announces failure if the event Forged ever happens (which it can easily check); (4) it simulates the de-signcryption oracle on u' by outputting $\perp$ on type (a) queries, and outputting the corresponding m' on type (b) queries (by using the table kept in rule (2)). It is clear than if Forged does not happen, $\mathcal{A}_2$ exactly simulates the behavior of $\mathcal{A}'$, and hence succeeds with overall probability at least $\frac{1}{2} + \frac{\varepsilon}{2}$.

C Proof of Theorem 3

The proof of IND-gCCA2-security is the same as in Theorem 1, since we did not use the security of signature there. Thus, we concentrate on showing UF-CMA-security. Let $\mathcal{A}'$ by PPT adversary trying to break UF-CMA-security of $\mathcal{S}t\mathcal{E}$ in the Outsider-security model. Recall, $\mathcal{A}'$ only knows $pub = (\mathsf{EK}_R, \mathsf{VK}_S)$, but has access to signcryption and the de-signcryption oracles SigEnc and VerDec. Let $m_1, \ldots, m_t$ be the queries $\mathcal{A}'$ asked the signcryption oracle, and $u_1, \ldots, u_t$ be the corresponding answers. Without loss of generality, we assume that $\mathcal{A}'$ never asks its de-signcryption oracle any query u' which is equivalent to one of the u_i's returned so far, i.e. where $\mathcal{R}'(u', u_i) = true$ (recall from Theorem 1 that the equivalence relation for $\mathcal{S}t\mathcal{E}$ was $\mathcal{R}' = \mathcal{R}$, i.e. the same as that for $\mathcal{E}$). Indeed, since $\mathcal{R}'$ can be evaluated by $\mathcal{A}'$, and $\mathcal{R}'$ is decryption-respecting, there is no need for $\mathcal{A}$ to ask such a query — 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 de-signcryption queries of $\mathcal{A}'$. Specifically, the signcryption query m_i is answered by generating $s_i \leftarrow \mathsf{Sig}_S(m_i)$ and returning $u_i \leftarrow \mathsf{Enc}_R(s_i)$. Let $\mathsf{Succ}_0(\mathcal{A}')$ be the success probability (i.e., that of forging a new signcryption) of $\mathcal{A}'$ in Env_0. Next, we define the following "hybrid" environments Env_j, $1 \leq j \leq t$. Each Env_j is identical to Env_0 above, except for one aspect: for the first j queries m_i $(1 \leq i \leq j)$ to the signcryption oracle, instead of returning $u_i \leftarrow \mathsf{Enc}_R(s_i)$, Env_i returns a random encryption of 0: $u_i \leftarrow \mathsf{Enc}_R(0)$. We let $\mathsf{Succ}_j(\mathcal{A}')$ be the success probability of $\mathcal{A}'$ in Env_j. Notice, Env_t answers all t queries "incorrectly".

We make two claims: (a) using the IND-gCCA2-security of $\mathcal{E}$, no PPT adversary $\mathcal{A}'$ can distinguish Env_{j-1} from Env_j, for any $1 \leq j \leq t$, i.e. $\mathsf{Succ}_{j-1}(\mathcal{A}') - \mathsf{Succ}_j(\mathcal{A}') \leq \mathsf{negl}(k)$; (2) using UF-NMA-security of $\mathcal{S}$, $\mathsf{Succ}_t(\mathcal{A}') \leq \mathsf{negl}(k)$, for any PPT $\mathcal{A}'$. Combined, claims (a) and (b) imply our theorem, since t is polynomial and $\mathsf{Succ}_0(\mathcal{A}') \leq (t+1) \cdot \mathsf{negl}(k) = \mathsf{negl}(k)$.

Proof of Claim (a). If for some $\mathcal{A}'$, $\mathsf{Succ}_{j-1}(\mathcal{A}') - \mathsf{Succ}_j(\mathcal{A}') > \varepsilon$ for non-negligible ε, we create $\mathcal{A}_1$ that will break the IND-gCCA2-security of $\mathcal{E}$ with probability ε. $\mathcal{A}_1$ views the keys of $\mathcal{E}$ as $(\mathsf{EK}_R, \mathsf{DK}_R)$, by itself picks a pair of signing keys $(\mathsf{SK}_S, \mathsf{VK}_S) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$, and hands $pub = (\mathsf{EK}_R, \mathsf{VK}_S)$ to $\mathcal{A}'$. $\mathcal{A}_1$ simulates all the de-signcryption queries u' of $\mathcal{A}'$ by using its own decryption oracle on u', and then validating the signature s it gets back, before returning the message $m = \mathsf{Msg}(s)$ to $\mathcal{A}'$. Simulation of the signcryption oracle is more intricate. $\mathcal{A}_1$ simulates the first $(j-1)$ signcryption queries u_i "incorrectly", by returning $u_i \leftarrow \mathsf{Enc}_R(0)$. At the j-th query m_j, $\mathcal{A}_1$ sets $s_j \leftarrow \mathsf{Sig}_S(m_j)$ and claims to distinguish the encryption of s_j from the encryption of 0. When given to distinguish an encryption (call it u_j for consistency) of either 0 or s_j, $\mathcal{A}_1$ hands u_j to $\mathcal{A}'$ as a signcryption of m_j. From that point on, all the remaining signcryption queries m_i are answered "correctly": $u_i \leftarrow \mathsf{Enc}_R(\mathsf{Sig}_S(m_i))$.

After $\mathcal{A}'$ returns a candidate forgery u, $\mathcal{A}_1$ checks if u is indeed a valid forgery by: (1) checking that u is not equivalent to any of the u_i's, including u_j; (2) using its decryption oracle on u, thus obtaining the presumed signature s; (3) validating that s is a valid signature of $m = \mathsf{Msg}(u)$; and (4) checking that $m \notin \{m_1 \ldots m_t\}$. If all of the above succeed, $\mathcal{A}_1$ guesses that the challenge u_j was the encryption of s_j (i.e., $\mathcal{A}'$ was run in Env_{j-1}), else it flips a random coin. We note that if u_j was the encryption of

s_j, we indeed ran $\mathcal{A}'$ in Env_{j-1}, otherwise, we ran it in Env_j. From the assumption, we get that the probability of $\mathcal{A}_1$ being correct is $\frac{1}{2} + \frac{\varepsilon}{2}$, a contradiction.

However, to complete the proof of claim (a), we also need to check that $\mathcal{A}_1$ never asked its decryption oracle to decrypt some u' equivalent (w.r.t. $\mathcal{R}$) to the challenge u_j. But we assumed that $\mathcal{A}'$ never asks its de-signcryption oracle a query u' equivalent (w.r.t. $\mathcal{R}'$) to any of the u_i's (in particular, u_j). Since $\mathcal{R}' = \mathcal{R}$ and $\mathcal{A}_1$ only uses the decryption oracles to answer de-signcryption queries of $\mathcal{A}'$ and to decrypt u, this is indeed so.

Proof of Claim (b). We note that in Env_t, the questions to the signcryption oracle are "useless": $\mathcal{A}'$ could have gotten the answers by itself by returning $\mathsf{Enc}_R(0)$. More formally, assuming $\mathcal{A}'$ forges a new signcryption with probability ε in Env_t, we can build a forger $\mathcal{A}_2$ for $\mathcal{S}$ that will contradict the UF-NMA-security of $\mathcal{S}$. $\mathcal{A}_2$ views the keys of $\mathcal{S}$ as $(\mathsf{SK}_S, \mathsf{VK}_S)$, by itself picks a pair of encryption keys $(\mathsf{EK}_R, \mathsf{DK}_R) \leftarrow \mathsf{Enc\text{-}Gen}(1^k)$, and hands $pub = (\mathsf{EK}_R, \mathsf{VK}_S)$ to $\mathcal{A}'$. From there on, $\mathcal{A}_2$ simulates the de-signcryption queries u' by getting $s' = \mathsf{Dec}_R(u')$ and returning $m' = \mathsf{Msg}(s')$ if $\mathsf{Ver}_R(s') = \mathsf{succeed}$. It also simulates the signcryption queries by returning $\mathsf{Enc}_R(0)$. When $\mathcal{A}'$ returns a forgery u, $\mathcal{A}_2$ outputs $s = \mathsf{Dec}(u)$. It is easy to see that $\mathcal{A}_2$ exactly recreates Env_t, completing the proof.

D Proof of Lemma 1

For one direction, we show that if $\mathcal{C}$ does not satisfy the hiding property, then $\mathcal{E}$ cannot even be IND-CPA-secure, let alone IND-gCCA2-secure. Indeed, if some adversary $\mathcal{A}$ can find m_0, m_1 s.t. $c(m_0) \not\approx c(m_1)$, then obviously $\mathsf{Enc}'(m_0) \equiv (c(m_0), \mathsf{Enc}(d(m_0))) \not\approx (c(m_1), \mathsf{Enc}(d(m_1))) \equiv \mathsf{Enc}'(m_1)$, contradicting IND-CPA-security.

Conversely, assume $\mathcal{C}$ satisfies the hiding property, and let $\mathcal{R}$ be the decryption-respecting equivalence relation w.r.t. which $\mathcal{E}$ is IND-CCA2-secure. We let the equivalence relation $\mathcal{R}'$ for $\mathcal{E}'$ be $\mathcal{R}'((c_1, e_1), (c_2, e_2)) = true$ iff $\mathcal{R}(e_1, e_2) = true$ and $c_1 = c_2$. It is easy to see that $\mathcal{R}'$ is decryption-respecting, since if $d_i = \mathsf{Dec}(e_i)$, then $\mathcal{R}'((c_1, e_1), (c_2, e_2)) = true$ implies that $(c_1, d_1) = (c_2, d_2)$, which implies that $m_1 = \mathsf{Open}(c_1, d_1) = \mathsf{Open}(c_2, d_2) = m_2$.

We now show IND-CCA2-security of $\mathcal{E}'$ w.r.t. $\mathcal{R}'$. For that, let Env_1 denote the usual environment where we place any adversary $\mathcal{A}'$ for $\mathcal{E}'$. Namely, (1) in find Env_1 honestly answers the decryption queries of $\mathcal{A}'$; (2) after m_0 and m_1 are selected, Env_1 picks a random b, sets $(c_b, d_b) \leftarrow \mathsf{Commit}(m_b)$, $e_b \leftarrow \mathsf{Enc}(d_b)$ and returns $\tilde{e} = \mathsf{Enc}'(m_b) = (c_b, e_b)$; (3) in guess, Env_1 honestly answers decryption query $e' = (c, e)$ provided $\mathcal{R}'(e', \tilde{e}) = false$. We can assume that $\mathcal{A}'$ never asks a query (c, e) where $\mathcal{R}(e, e_b) = true$ but $c \neq c_b$. Indeed, by our assumption only the value $c = c_b$ will check with d_b, so the answer to queries with $c \neq c_b$ is $\perp$ (and $\mathcal{A}'$ knows it). Hence, we can assume that $\mathcal{R}'(e', \tilde{e}) = false$ implies that $\mathcal{R}'(e, e_b) = false$. We let $\mathsf{Succ}_1(\mathcal{A}')$ denote the probability $\mathcal{A}'$ succeeds in predicting b. Then, we define the following "fake" environment Env_2. It is identical to Env_1 above, except for one aspect: in step (2) it would return bogus encryption $\tilde{e} = (c(0), e_b)$, i.e. puts the commitment to the zero string 0 instead of the expected c_b. In particular, step (3) is the same as before with the understanding that $\mathcal{R}'(e', \tilde{e})$ is evaluated with the fake challenge $\tilde{e}$. We let $\mathsf{Succ}_2(\mathcal{A}')$ be the success probability of $\mathcal{A}$ in Env_2.

We make two claims: (a) using the hiding property of $\mathcal{C}$, no PPT adversary $\mathcal{A}'$ can distinguish Env_1 from Env_2, i.e. $|\mathsf{Succ}_1(\mathcal{A}') - \mathsf{Succ}_2(\mathcal{A}')| \leq \mathsf{negl}(k)$; (b) using IND-gCCA2-security of $\mathcal{E}$, $\mathsf{Succ}_2(\mathcal{A}') < \frac{1}{2} + \mathsf{negl}(k)$, for any PPT $\mathcal{A}'$. Combined, claims (a) and (b) imply the lemma.

Proof of Claim (a). If for some $\mathcal{A}'$, $\mathsf{Succ}_1(\mathcal{A}') - \mathsf{Succ}_2(\mathcal{A}') > \varepsilon$ for non-negligible ε, we create $\mathcal{A}_1$ that will break the hiding property of $\mathcal{C}$. $\mathcal{A}_1$ picks $(\mathsf{EK}, \mathsf{DK}) \leftarrow \mathsf{Enc\text{-}Gen}(1^k)$ by itself, and runs $\mathcal{A}'$

(answering its decryption queries using DK) until $\mathcal{A}'$ outputs m_0 and m_1. At this stage $\mathcal{A}_1$ picks a random $b \leftarrow \{0,1\}$, and claims to be able to distinguish $c(0)$ from $c_b = c(m_b)$. When presented with $\tilde{c}$ — a commitment to either 0 or m_b — $\mathcal{A}_1$ will return to $\mathcal{A}'$ the "ciphertext" $\tilde{e} = (\tilde{c}, e_b)$. $\mathcal{A}_1$ will then again run $\mathcal{A}'$ to completion refusing to decrypt e' such that $\mathcal{R}'(e', \tilde{e}) = true$. When $\mathcal{A}'$ outputs $\tilde{b}$, $\mathcal{A}_1$ says that the message was m_b if $\mathcal{A}'$ succeeds ($\tilde{b} = b$), and says 0 otherwise. It is easy to check that in case $\tilde{c} = c(m_b) = c_b$, $\mathcal{A}'$ was run exactly in Env_1, otherwise — in Env_2, which easily implies that $\Pr(\mathcal{A}_1 \text{ succeeds}) \geq \frac{1}{2} + \frac{\varepsilon}{2}$, a contradiction.

Proof of Claim (b). If for some $\mathcal{A}'$, $\mathsf{Succ}_2(\mathcal{A}') > \frac{1}{2} + \varepsilon$, we create $\mathcal{A}_2$ which will break IND-gCCA2-security of $\mathcal{E}$. Specifically, $\mathcal{A}_2$ can simulate the decryption query $e' = (c, e)$ of $\mathcal{A}'$ by asking its own decryption oracle to decrypt $d = \mathsf{Dec}(e)$, and returning $\mathsf{Open}(c, d)$. When $\mathcal{A}'$ outputs m_0 and m_1, $\mathcal{A}_2$ sets $(c_i, d_i) \leftarrow \mathsf{Commit}(m_i)$ and claims to distinguish d_0 and d_1. When given challenge $e_b \leftarrow \mathsf{Enc}(d_b)$ for unknown b, $\mathcal{A}_2$ gives $\mathcal{A}'$ the challenge $\tilde{e} = (c(0), e_b)$. Then, again, $\mathcal{A}_2$ uses its own decryption oracle to answer all queries $e' = (c, e)$ as long as $\mathcal{R}'(e', \tilde{e}) = false$. From the definition of $\mathcal{R}'$ and our assumption earlier, we see that $\mathcal{R}(e, e_b) = false$ as well, so all such queries are legal. Since $\mathcal{A}_2$ exactly recreates the environment Env_2 for $\mathcal{A}'$, $\mathcal{A}_2$ succeeds with probability $\mathsf{Succ}_2(\mathcal{A}') > \frac{1}{2} + \varepsilon$.

E Proof of Lemma 2

For one direction, we show that if $\mathcal{C}$ does not satisfy the relaxed binding property, then $\mathcal{S}'$ cannot be UF-CMA-secure. Indeed, assume for some adversary $\mathcal{A}$ can produce m such that when $(c, d) \leftarrow \mathsf{Commit}(m)$ is generated and given to $\mathcal{A}$, $\mathcal{A}$ can find (with non-negligible probability ε) a value d' such that $\mathsf{Open}(c, d') = m'$ and $m' \neq m$. We build a forger $\mathcal{A}'$ for $\mathcal{S}'$ using $\mathcal{A}$. $\mathcal{A}'$ gets m from $\mathcal{A}$, and asks its signing oracle to sign m. $\mathcal{A}'$ gets back (s, d), where s is a valid signature of c, and (c, d) is a random commitment pair for m. $\mathcal{A}'$ gives (c, d) to $\mathcal{A}$, and gets back (with probability ε) the value d' such that $\mathsf{Open}(c, d') = m'$ different from m. But then (s, d') is a valid signature (w.r.t. $\mathcal{S}'$) of a "new" message m', contradicting the UF-CMA-security of $\mathcal{S}$.

Conversely, assume some forger $\mathcal{A}'$ breaks the UF-CMA-security of $\mathcal{S}'$ with non-negligible probability ε. Assume $\mathcal{A}'$ made (wlog exactly) $t = t(k)$ oracle queries to Sig' for some polynomial $t(k)$. For $1 \leq i \leq t$, we let m_i be the i-th message $\mathcal{A}'$ asked to sign, and (s_i, d_i) be its signature (where $(c_i, d_i) \leftarrow \mathsf{Commit}(m_i)$ and $s_i \leftarrow \mathsf{Sig}(c_i)$). We also let m, s, d, c have similar meaning for the message that $\mathcal{A}'$ forged. Finally, let Forged denote the event that $c \notin \{c_1, \ldots, c_t\}$. Notice,

$$\varepsilon < \Pr(\mathcal{A}' \text{ succeeds}) = \Pr(\mathcal{A}' \text{ succeeds } \wedge \ \mathsf{Forged}) + \Pr(\mathcal{A}' \text{ succeeds } \wedge \ \overline{\mathsf{Forged}})$$

Thus, at least one of the probabilities above is $\geq \varepsilon/2$. We show that the first case contradicts the UF-CMA-security of $\mathcal{S}$, while the second case contradicts the relaxed binding property of $\mathcal{C}$.

Case 1: $\Pr(\mathcal{A}' \text{ succeeds } \wedge \mathsf{Forged}) \geq \varepsilon/2$. We construct a forger $\mathcal{A}_1$ for $\mathcal{S}$. It simulates the run of $\mathcal{A}'$ by generating a commitment key CK by itself, and using its own signing oracle to answer the signing queries of $\mathcal{A}'$: set $(c_i, d_i) \leftarrow \mathsf{Commit}(m_i)$, get $s_i \leftarrow \mathsf{Sig}'(c_i)$ from the oracle, and return (s_i, d_i). When $\mathcal{A}'$ forges a signature (s, d) of m w.r.t. $\mathcal{S}'$, $\mathcal{A}_1$ forges a signature s of c w.r.t. $\mathcal{S}$. Notice, c is a "new forgery" in $\mathcal{S}$ iff Forged happens. Hence, $\mathcal{A}_1$ succeeds with probability at least $\varepsilon/2$, a contradiction to UF-CMA-security of $\mathcal{S}$.

Case 2: $\Pr(\mathcal{A}' \text{ succeeds } \wedge \ \overline{\mathsf{Forged}}) \geq \varepsilon/2$. We construct an adversary $\mathcal{A}_2$ contradicting the relaxed binding property of $\mathcal{C}$. $\mathcal{A}_2$ will generate its own key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$, and will also pick a random index $1 \leq i \leq t$. It simulates the run of $\mathcal{A}'$ in a standard manner (same way as $\mathcal{A}_1$ above) up to the point where $\mathcal{A}'$ asks its i-th query m_i. At this stage $\mathcal{A}_2$ outputs m_i as its output to the find stage.

When receiving back random $(c_i, d_i) \leftarrow \mathsf{Commit}(m_i)$, it uses them to sign m_i as before (i.e., returns $(\mathsf{Sig}(c_i), d_i)$ to $\mathcal{A}'$), and keeps simulating the run of $\mathcal{A}'$ in the usual manner. When $\mathcal{A}$ outputs the forgery (s, d) of a message m, $\mathcal{A}_2$ checks if $c_i = c$ ($\mathsf{Msg}(s)$) and $m_i \neq m$. If this fails, it fails as well. Otherwise, it outputs d as its final output to the $\mathsf{collide}$ stage. We note that when Forged does not happen, i.e. $c \in \{c_1 \ldots c_t\}$, we have $c = c_i$ with probability at least $1/t$. Thus, with overall non-negligible probability $\varepsilon/(2t)$ we have that: (1) $m \neq m_i$ ($\mathcal{A}'$ outputs a new message m); (2) $c_i = c$ (Forged did not happen and $\mathcal{A}_2$ correctly guessed i such that $c_i = c$); (3) $\mathsf{Open}(c, d) = m$ and $\mathsf{Open}(c, d_i) = m_i$. But this exactly means that $\mathcal{A}_2$ broke the relaxed binding property of $\mathcal{C}$, a contradiction.

F Summary of Implications for CCA2/sUF Notions

We indicate which of our results for signcryption (stated for gCCA2/UF notions) extend to the CCA2/sUF notions.

Encrypt-then-sign $\mathcal{E}t\mathcal{S}$. $\mathcal{E}t\mathcal{S}$ signcryption preserves sUF unforgeability for signatures (since the signature is applied on the outside layer). However, $\mathcal{E}t\mathcal{S}$ does not in general preserve the CCA2-security for the encryption. In particular, CCA2-security is not preserved in the Insider-security model as long as the signature is probabilistic. Indeed, the sender S can always regenerate a new signature of $e = \mathsf{Msg}(u)$, when given the challenge u to de-signcrypt. In the Outsider-security model, CCA2-security is not necessarily preserved as well, provided the signature $\mathcal{S}$ is not strongly unforgeable. Indeed, if $\mathcal{S}$ allows the adversary to obtain a different signature of the *same* message (e.g., the last bit of the signature is "useless"), then the adversary can decrypt $u = \mathsf{Sig}_S(e)$ by simply changing u to a different signature u' of e, and asking its oracle to de-signcrypt u' (which is now artificially "different" from the challenge u). On a positive note, if $\mathcal{S}$ is sUF-CMA-secure, then we easily obtain the analog of Theorem 2 in the Outsider-model: $\mathcal{E}t\mathcal{S}$ *amplifies* the security of $\mathcal{E}$ from CPA to CCA2 level.

Sign-then-encrypt $\mathcal{S}t\mathcal{E}$. The results get completely "reversed" compared with the $\mathcal{E}t\mathcal{S}$ signcryption. Now, IND-CCA2-security is preserved (since the encryption is applied on the outside layer). However, $\mathcal{S}t\mathcal{E}$ does not in general preserve the sUF-CMA-security for the signature. In particular, sUF-CMA-security is *never* preserved in the Insider-security model since the encryption must be probabilistic. Indeed, the receiver R can always regenerate a new encryption of $s = \mathsf{Dec}_R(u)$, resulting in a new signcryption of the same message. In the Outsider-security model, sUF-CMA-security is not necessarily preserved as well, provided the equivalence relation of the IND-gCCA2-secure encryption scheme $\mathcal{E}$ is *not* the identity and $\mathcal{E}$ allows the adversary to obtain a different encryption of the *same* message (e.g., the last bit of the encryption is "useless"). Indeed, then the adversary can forge a new signcryption of the same message by simply changing $u = \mathsf{Enc}_R(s)$ to a different encryption u' of s. On a positive note, if $\mathcal{E}$ is IND-CCA2-secure, then we easily obtain the analog of Theorem 3 in the Outsider-model: $\mathcal{S}t\mathcal{E}$ *amplifies* the security of $\mathcal{S}$ from UF-NMA to sUF-CMA level.

Commit-then-encrypt-and-sign $\mathcal{C}t\mathcal{E}\&\mathcal{S}$. First, the auxiliary Lemma 1 about $\mathsf{Enc}'(m) = (c, \mathsf{Enc}(d))$ still holds. Indeed, the equivalence relation $\mathcal{R}'$ remains the equality relation when $\mathcal{R}$ is the equality relation. On the other hand, Lemma 2 about $\mathsf{Sig}'(m) = (\mathsf{Sig}(c), d)$ does not hold as stated, since relaxed binding leaves open the possibility of finding a value $d' \neq d$ such that $\mathsf{Open}(c, d) = \mathsf{Open}(c, d')$ (say, the last bit of d is "useless"). However, if we strengthen the relaxed binding property to say that one cannot even find a different commitment pair (c, d') to the same message, then sUF-CMA-security will be preserved in Lemma 2. In any event, despite these results for auxiliary lemmas, the analog of Theorem 4 does not hold in the Insider-security model for either IND-CCA2 or sUF-CMA, since signatures and encryptions are usually probabilistic.