On Generic Constructions of Designated Confirmer Signatures - II

(The "Signature of a Commitment" Paradigm Revisited)

Laila El Aimani

b-it, Dahlmannstr. 2, Universität Bonn, 53113 Bonn, Germany
elaimani@bit.uni-bonn.de

Abstract. Generic constructions of designated confirmer signatures follow one of the following two strategies; either produce a digital signature on the message to be signed, then encrypt the resulting signature, or produce a commitment on the message, encrypt the string used to generate the commitment and finally sign the latter. We study the second strategy by determining the exact security property needed in the encryption to achieve secure constructions. This study infers the exclusion of a useful type of encryption from the design due an intrinsic weakness in the paradigm. Next, we propose a simple method to remediate to this weakness and we get efficient constructions which can be used with *any* digital signature.

Keywords: Designated Confirmer signatures, "Signature of a commitment" paradigm, Generic construction, Reduction/meta-reduction, Zero Knowledge.

1 Introduction

Digital signatures were introduced in [13] as an analogous to signatures in the paper world to seize most properties needed in a signature, for instance, the universal verification. However, in some applications, the signer might want to restrain the holder of a signature from convincing other parties of the validity of the signature in question. A typical illustration of such a need is this real-life scenario from [19]. An employer issues a job offer to a certain candidate. Naturally, the employer needs to compete with the other job offers in order to attract the good candidate. Therefore, he does not wish the offer to be revealed to his competitors. At the same time, the candidate needs more than a verbal or unsigned agreement in order to protect himself from the employer not keeping his promise. Undeniable signatures, introduced in [10], provide a good solution to this problem as they are: 1. only verified with the help of the signer, 2. non transferable, 3. binding in the sense that a signer cannot deny a signature he has actually issued. The only drawback of these signatures is that unavailability of the signer obstructs the entire process. To overcome this problem, designated confirmer signatures were introduced in [8], where the confirmation/denial of a signature is delegated to a *designated confirmer*. With this solution, the signer can confirm only signatures he has just generated, whilst the confirmer can confirm/deny any signature. Actually, in the literature, there is a clear separation between confirmer signatures and *directed signatures* [23], which share the same concept as confirmer signatures with the exception of allowing both the signer and the confirmer to confirm/deny signatures. Finally, a desirable property in designated confirmer signatures is the convertibility of the signatures to ordinary ones. Indeed, looking at the job scenario, it would be preferable to be able to convert the contract of the candidate, once he officially joins the company, to a universally verifiable one instead of having to issue a new contract.

1.1 Related work

Since the introduction of confirmer signatures, a number of attempts have been made to produce them from basic primitives. Most such proposals fall into one of the following two categories:

"Encryption of a signature" approach. This approach consists in first producing a digital signature on the message to be signed, then encrypting the produced signature using a suitable cryptosystem. The construction was first formally [1] described in [6], and required the components to meet the highest security notions (EUF-CMA signatures and IND-CCA encryption). The main weakness of the construction lies in the resort to concurrent zero knowledge (ZK) protocols of general NP statements in the confirmation/denial protocol. Later, the construction in [19] managed to circumvent the problem by encrypting the digital signature during the confirmation protocol. With this trick, the authors managed to get rid of concurrent ZK proofs of general NP statements in the confirmation protocol (the denial protocol still suffers the recourse to such proofs), but at the expense of the security and the length of the resulting signatures. Another construction implementing this principle is given in [11]; the construction uses cryptosystems with labels and is analyzed in a more elaborate security model. However, it is supplied with only one efficient instantiation as the confirmation/denial protocols still resort to concurrent ZK protocols of general NP statements. Finally, the last proposal in this category is given in [22], where we propose a construction using certain cryptosystems that are required to be only IND-CPA secure. As a consequence, we manage to get efficient confirmation/denial protocols in case the construction is instantiated from a specific class of signature schemes (similar to the one considered in [19]). Moreover, the resulting confirmer signatures are very efficient (small generation/verification/conversion cost and short signatures due to IND-CPA encryption) and they enjoy a maximal security. However, although the considered class of digital signatures includes most proposals that appeared in the literature, there exists some schemes which do not seem to belong to it, e.g., the PSS signature scheme [2].

"Signature of a commitment" approach. This technique consists in generating a commitment on the message to be signed, then signing the produced commitment using a digital signature scheme. The confirmer signature is comprised of both the commitment and the signature. The first proposal that realizes this principle is [25] where a construction of confirmer signatures from digital signatures obtained from the Fiat-Shamir paradigm is presented. Thus, the resulting confirmer signatures can be only proven secure in the random oracle model (ROM), inheriting this property from the use of the Fiat-Shamir paradigm, which constitutes their major shortcoming. Actually, it is well known, according to [32], that most discrete-logarithm-based signatures obtained from the Fiat-Shamir technique are very unlikely to preserve the same level of security in the standard model. Moreover, the construction does not support the conversion of the confirmer signatures. In [16] and [35], a construction which supports the conversion of the signatures and applies to *any* digital signature scheme was proposed. The key idea behind the proposal resides in augmenting the confirmer signature (comprised of the commitment and a signature on it) by the encryption of the random string used to generate the commitment. Although the confirmation/denial protocols involve general ZK proofs since the confirmer has to prove in concurrent ZK the knowledge of the decryption of an IND-CCA encryption and of a string used for commitment, the construction accepts an efficient instantiation using Camenisch-Shoup's verifiable encryption scheme [7] and Pedersen's commitment scheme.

To finish the exhaustive list of constructions of confirmer signatures, we must cite the first construction due to Okamoto [28], which was used to prove equivalence between confirmer signatures and public key encryption with respect to existence. Thus, efficiency was not taken into account in the framework. There is also the construction [26] which uses an undeniable signature among its building blocks and provides a restricted security (under lunch time attacks) in the ROM. Finally, In [35], the authors proposed a second construction which does not require any encryption, but at the expense of the underlying security assumption. In fact, it has its invisibility resting on the decisional Diffie-Hellman assumption, which rules out using the scheme in bilinear groups and thus benefiting from the attractive features they present such as achieving short group elements. Moreover, the construction suffers also the recourse to the ROM.

In this paper, we revisit the second approach, namely the "signature of a commitment" method. In fact, efficient as the first approach is, it still applies only to a restricted class of signatures. This is clearly

[1] The idea without proof was already known, for instance, it was mentioned in [12].

manifested in the constructions in [19] or [22] which do not seem to be plausible with the signature PSS [2]. Our goal is to further improve the "commit then sign" method in terms of efficiency (signature length and cost) and security by allowing more efficient instantiations of the encryption and commitment schemes used as building blocks.

1.2 Contributions and key ideas

We make three contributions. First, we revisit the constructions implementing the "signature of a commitment" paradigm, namely those provided in [16, 35]. We prove that indistinguishable cryptosystems under a *plaintext checking attack* (IND-PCA secure) are necessary and sufficient to obtain secure confirmer signatures. Our approach is similar to the one provided in [22] to study the "encryption of a signature" technique. In fact, we first exclude non malleable encryption from the design (NM-CPA secure cryptosystems), which rules out the weaker notions that are IND-CPA and OW-CPA. We do this by means of an efficient tool, called meta-reductions, which was used in a number of important cryptographic results [4, 32, 31, 30]. Then, we exclude the OW-CCA notion by a similar technique, which again rules out the OW-PCA notion. The notion that has to be considered next is IND-PCA which luckily turns out to be sufficient to achieve secure constructions. We conclude that, although we mange to weaken the assumption on the encryption (from IND-CCA as claimed in [16, 35] to IND-PCA), the construction still cannot allow homomorphic encryption in the design since a homomorphic cryptosystem can never be IND-PCA secure. This is unfortunate since such encryption proved to be efficient decryption verifiable (see [22] for an illustration), i.e., possesses efficient ZK protocols for proving the knowledge of the plaintext underlying a given ciphertext, and such a property in profoundly needed in the confirmation/denial protocols.

In the second contribution, we tackle the problem of homomorphic encryption in the design; we show that using a small trick that consists in producing the digital signature on the commitment *concatenated with the encryption of the string used in the commitment* suffices to make the security needed in the encryption drop drastically to being only IND-CPA secure. The key idea is to remark that the original construction is not strongly unforgeable, i.e., one can produce a valid confirmer signature without the help of the signer, which explains the need for a decryption or a plaintext checking oracle (CCA or PCA security) to handle such signatures. With the small trick, we are able to annihilate this weakness and allow a weak encryption in the design without compromising the overall security. As a result, we achieve better performances that manifest in a short signature, a small signature generation, verification and conversion cost, and finally more efficient instantiations of the construction (instead of using only Camenisch-Shoup encryption and Pedersen commitment) by allowing homomorphic encryption.

Finally, our last contribution sheds light on a particular sub-case of the "signature of a commitment" paradigm, which consists in using IND-CPA encryption instead of the commitment scheme. In fact, it is well known that IND-CPA encryption yields secure commitment schemes, which makes such an instantiation plausible. However, the bright side of this technique consists in not requiring the encryption of the random string anymore. Thus, a confirmer signature on a given message can be achieved by encrypting the message to be signed, then producing a digital signature on this encryption. The pair consisting of the encryption and the resulting signature forms the confirmer signature on the message. This method clearly improves the original paradigm, however it necessitates efficient non-interactive proofs of knowledge. This is no longer a problem nowadays due to the progress of research made recently in this area, e.g., [21].

2 Convertible Designated Confirmer Signatures (CDCS)

Since their introduction, many definitions and security models for CDCS have emerged. We adhere to the following model which implies many popular models for CDCS, for instance, the one adopted in [16, 35], where constructions from the "signature of a commitment" method were provided.

We refer to Appendix A for the comparison of the present model with the other models, as well as for the necessary cryptographic primitives that will come into use, that are, digital signatures, public key encryption schemes, commitment schemes, and finally Σ protocols.

2.1 Syntax

A CDCS scheme consists of the following procedures:

Key generation. Generates probabilistically key pairs $(\mathsf{sk}_S, \mathsf{pk}_S)$ and $(\mathsf{sk}_C, \mathsf{pk}_C)$ for the signer and for the confirmer respectively, consisting of the private and the public key.

ConfirmedSign. On input sk_S, pk_C and a message m, outputs a confirmer signature signature μ, then interacts with the signature recipient to convince him of the validity of the just generated signature.

Confirmation/Denial protocol. These are interactive protocols between the confirmer and a verifier. Their common input consists of, in addition to pk_S and pk_C, the alleged signature μ, and the message m in question. The confirmer uses his private key sk_C to convince the verifier of the validity (invalidity) of the signature μ on m. At the end, the verifier either accepts or rejects the proof.

Selective conversion. This is an algorithm run by the confirmer using sk_C, in addition to pk_C and pk_S. The result is either $\perp$ or a string which allows the signature to be universally verified as a valid digital signature.

Selective verification. This is an algorithm for verifying converted signatures. It inputs the converted signature, the message and pk_S and outputs either 0 or 1.

Remark 1. In [16, 35], the authors give the possibility of obtaining *directly* digital signatures on a given message. We find this unnecessary since it is already enough that a CDCS scheme supports the convertibility feature. Moreover, in [11], the author considers a further protocol used by the confirmer to prove the correctness of the conversion. We will see that all the constructions provided in this paper extend readily to this augmented model.

2.2 Security model.

The above algorithms must be complete. Moreover the confirmedSign, confirmation and denial protocols must be complete, sound and zero knowledge. In the sequel, we describe two further properties that a CDCS scheme should meet.

Security for the signer (unforgeability). It is defined through the following game: the adversary $\mathcal{A}$ is given the public parameters of the CDCS scheme, namely pk_S and pk_C, the public key of the signer and of the confirmer resp, in addition to the private key sk_C of the confirmer. $\mathcal{A}$ is further allowed to query the signer on polynomially many messages, say q_s. At the end, $\mathcal{A}$ outputs a pair consisting of a message m, that has not been queried yet, and a string μ. $\mathcal{A}$ wins the game if μ is a valid confirmer signature on m. We say that a CDCS scheme is (t, ϵ, q_s)-EUF-CMA secure if there is no adversary, operating in time t, that wins the above game with probability greater than ϵ.

Security for the confirmer (invisibility). Invisibility against a chosen message attack (INV1-CMA) is defined through the following game between an attacker $\mathcal{A}$ and his challenger $\mathcal{R}$: after $\mathcal{A}$ gets the public parameters of the scheme from $\mathcal{R}$, he starts **Phase 1** where he queries the signing, confirmation/denial, selective conversion oracles in an adaptive way. Once $\mathcal{A}$ decides that **Phase 1** is over, he outputs two messages m_0, m_1 that have not been queried before to the signing oracle and requests a challenge signature $\mu^\star$. $\mathcal{R}$ picks uniformly at random a bit $b \in \{0,1\}$. Then $\mu^\star$ is generated using the signing oracle on the message m_b. Next, $\mathcal{A}$ starts adaptively querying the previous oracles (**Phase 2**), with the exception of not querying m_0, m_1 to the signing oracle and $(m_i, \mu^\star)$, $i = 0, 1$, to the confirmation/denial and selective conversion oracles. At the end, $\mathcal{A}$ outputs a bit b'. He wins the game if $b = b'$. We define $\mathcal{A}$'s advantage as $\mathsf{adv}(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$. We say that a CDCS scheme is $(t, \epsilon, q_s, q_v, q_{sc})$-INV1-CMA secure if no adversary operating in time t, issuing q_s queries to the signing oracle, q_v queries to the confirmation/denial oracles and q_{sc} queries to the selective conversion oracle wins the above game with advantage greater that ϵ.

3 The Plain "Signature of a Commitment" Paradigm

The first construction of CDCS that realizes the "Signature of a Commitment" principle was given in [16], then it was refined in [35]. This construction devises a convertible confirmer signature (CDCS) using an EUF-CMA signature scheme $\varSigma$, an IND-CCA encryption scheme $\varGamma$ with labels and a secure commitment scheme $\varOmega$. The signer key pair consists of $(\varSigma.\mathsf{pk}, \varSigma.\mathsf{sk})$, corresponding to the key pair of the signature scheme $\varSigma$, whereas the confirmer key pair consists of $(\varGamma.\mathsf{sk}, \varGamma.\mathsf{sk})$ which corresponds to the key pair related to $\varGamma$. To sign a message m, the signer first computes a commitment c on the message, then encrypts, under the label $m\|\varSigma.\mathsf{pk}$, the random string used for the commitment, say r, and finally, signs the commitment c using $\varSigma.\mathsf{sk}$. The confirmer signature consists of the triple $(\varGamma.\mathsf{encrypt}_{\varGamma.\mathsf{pk}, m\|\varSigma.\mathsf{pk}}(r), c = \varOmega.\mathsf{commit}(m, r), \varSigma.\mathsf{sign}_{\varSigma.\mathsf{sk}}(c))$. To confirm/deny a signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a given message m, the confirmer first checks whether μ_3 is a valid digital signature on μ_2 w.r.t. $\varSigma.\mathsf{pk}$, if so, he provides a concurrent ZK proof (using his private key $\varGamma.\mathsf{sk}$) of the equality/inequality of the decryption of μ_1 and the opening value of the commitment μ_2 w.r.t. m. Such a proof is plausible since the encryption and commitment algorithms in a cryptosystem and a commitment scheme resp define an NP (co-NP in case of inequality) language that accepts a zero knowledge proof system. Note that the signer can also provide such a proof in case the alleged signature has just been generated (using the randomness used to generate the encryption μ_1). Selective conversion of a signature $\mu = (\mu_1, \mu_2, \mu_3)$ is achieved by releasing the decryption of μ_1, in case μ is valid, or the symbol $\perp$ otherwise.

Completeness, soundness and non-transferability of the confirmedSign, confirmation/denial protocols follow directly by using zero knowledge proofs of knowledge. Concerning unforgeability of the resulting confirmer signatures, it rests on the EUF-CMA security and on the binding property of the underlying digital signature scheme and commitment scheme resp. Finally, invisibility (INV1-CMA) is attained by using an IND-CCA secure cryptosystem with labels and a secure commitment scheme. Details about the proofs were not given so far, but are due to appear in a forthcoming paper (full version of [35]). Since the paper is not available yet, we flesh out the proofs in Appendix B.

In the rest of this section, we prove that IND-PCA cryptosystems with labels are necessary and sufficient to obtain invisible signatures. We must note here that using cryptosystems with labels was suggested in [35] to provide invisibility of the resulting signatures in a model where the attacker can directly obtain *digital signatures* on any message of his choice. As mentioned in Subsection 2.1, we do not opt for this model since it is unnecessary, however cryptosystems with labels proved to be requisite for the analysis that will follow.

3.1 The exact invisibility of the construction

In this subsection, we prove that IND-PCA cryptosystems with labels are necessary and sufficient to achieve invisible signatures. Our study is similar to the one provided in [22] which analyzes the plain "encryption of a signature" paradigm. Thus, we will first exclude NM-CPA secure cryptosystems with labels from use, which will rule out automatically IND-CPA and OW-CPA cryptosystems. We do this using an efficient algorithm (a *meta-reduction*) which transforms an algorithm (*reduction*), reducing the invisibility of the confirmer signatures to the NM-CPA security of the underlying cryptosystem, to an algorithm breaking the NM-CPA security of the same cryptosystem. Hence, such a result suggests that under the assumption of the underlying cryptosystem being NM-CPA secure, there exists no such a reduction, or if it (the cryptosystem) is not NM-CPA secure, such a reduction will be useless. Next, we exclude similarly OW-CCA cryptosystems from the design. The next security notion that has to be considered is IND-PCA, which turns out to be sufficient to achieve invisibility. Likewise, our impossibility results are in a first stage partial in the sense that they apply only to *key preserving* reductions, i.e., reductions which, trying to attack a property of a cryptosystem given by the public key pk, feed the invisibility adversary with the confirmer public key pk. Next, we extend the result to arbitrary reductions under some complexity assumption on the cryptosystem in question.

Lemma 1. *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an INV1-CMA adversary $\mathcal{A}$ against the above construction to an NM-CPA adversary against the underlying cryptosystem. Then, there exists a meta-reduction $\mathcal{M}$ that NM-CPA breaks the cryptosystem in question.*

As mentioned in the discussion above, the lemma claims that under the assumption of the underlying cryptosystem being NM-CPA secure, there exists no key-preserving reduction $\mathcal{R}$ that reduces NM-CPA breaking the cryptosystem in question to INV1-CMA breaking the construction, or if there exists such an algorithm, the underlying cryptosystem is not NM-CPA secure, thus rendering such a reduction useless.

Proof. Let $\mathcal{R}$ be a key-preserving reduction that reduces the invisibility of the construction to the NM-CPA security of its underlying cryptosystem. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to NM-CPA break the same cryptosystem by simulating an execution of the INV1-CMA adversary $\mathcal{A}$ against the construction.

Let $\mathit{\Gamma}$ be the cryptosystem with labels $\mathcal{M}$ is trying to attack. $\mathcal{M}$ launches $\mathcal{R}$ over $\mathit{\Gamma}$ with the same public key, say $\mathit{\Gamma}.\mathsf{pk}$. $\mathcal{M}$, acting as the INV1-CMA adversary against the construction, queries $\mathcal{R}$ on $m_0, m_1 \overset{R}{\longleftarrow} \{0,1\}^\star$ for confirmer signatures. Then he queries the resulting strings $\mu_0 = (\mu_0^1, \mu_0^2, \mu_0^3)$ and $\mu_1 = (\mu_1^1, \mu_1^2, \mu_1^3)$ (corresponding to the confirmer signatures on m_0 and m_1 respectively) for a selective conversion. Let r_0 and r_1 be the output decryption of μ_0^1 and μ_1^1 resp (i.e., the randomnesses used generate the commitments μ_0^2 and μ_1^2 on m_0 and m_1 resp). With overwhelming probability, we have $r_0 \neq r_1$ [2], and if it is not the case, $\mathcal{M}$ will repeat the experiment until he obtains two different r_0 and r_1. Then, $\mathcal{M}$ inputs $\mathcal{D} = \{r_0, r_1\}$ to his own challenger as a distribution probability from which the plaintexts will be drawn. Moreover, he chooses uniformly at random a bit $b \overset{R}{\longleftarrow} \{0,1\}$ and outputs to his challenger the challenge label $m_b \| \mathit{\Sigma}.\mathsf{pk}$, where $\mathit{\Sigma}.\mathsf{pk}$ is the public key of the digital signature underlying the construction. $\mathcal{M}$ will receive as a challenge encryption $\mu_b^\star$. At that point, $\mathcal{M}$ will query $\mathcal{R}$ on the string $(\mu_b^\star, \mu_b^2, \mu_b^3)$ and the message m_b for a selective conversion. If the result of such a query is different from $\perp$, then, $\mu_b^\star$ is a valid encryption of the random string used to generate the commitment μ_b^2, namely r_b. $\mathcal{M}$ will then output to his challenger an encryption μ of $\overline{r_b}$ under the same challenge label $m_b \| \mathit{\Sigma}.\mathsf{pk}$, where $\overline{r_b}$ refers to the bit-complement of the element r_b, and the relation R: $R(r, r') = (r' = \overline{r})$. Otherwise, he will output an encryption of $\overline{r_{1-b}}$ (under the same challenge label) and the same relation R. Finally $\mathcal{M}$ aborts the game (stops simulating an INV1-CMA attacker against the generic construction). □

Lemma 2. *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an INV1-CMA adversary $\mathcal{A}$ against the above construction into a OW-CCA adversary against the underlying cryptosystem. Then, there exists a meta-reduction $\mathcal{M}$ that OW-CCA breaks the cryptosystem in question.*

Proof. The proof technique is similar to the one above. Let $\mathcal{R}$ be the key-preserving reduction that reduces the invisibility of the construction to the OW-CCA security of the underlying cryptosystem. We construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to OW-CCA break the same cryptosystem by simulating an execution of the INV1-CMA adversary $\mathcal{A}$ against the construction.

Let $\mathit{\Gamma}$ be the cryptosystem $\mathcal{M}$ is trying to attack w.r.t. a public key $\mathit{\Gamma}.\mathsf{pk}$. $\mathcal{M}$ launches $\mathcal{R}$ over $\mathit{\Gamma}$ with the same public key $\mathit{\Gamma}.\mathsf{pk}$. After $\mathcal{M}$ gets the label L on which $\mathcal{R}$ wishes to be challenged, he ($\mathcal{M}$)

[2] Actually, if $\mathcal{R}$ uses always the same string to produce the commitments, then the construction is clearly not invisible.

forwards it to his own challenger. Finally, $\mathcal{M}$ gets a challenge ciphertext c, that he forwards to $\mathcal{R}$. Note that $\mathcal{M}$ is allowed to query the decryption oracle on any pair (ciphertext,label) except on the pair (c, L). Thus, all decryption queries made by $\mathcal{R}$, which are by definition different from the challenge (c, L), can be forwarded to $\mathcal{M}$'s own challenger. At some point, $\mathcal{M}$, acting as an INV1-CMA attacker against the construction, will output two messages m_0, m_1 such that $L \notin \{m_0 \| \Sigma.\mathsf{pk}, m_1 \| \Sigma.\mathsf{pk}\}$, where $\Sigma.\mathsf{pk}$ is the public key of the digital signature underlying the construction. $\mathcal{M}$ gets as response a challenge signature $\mu^\star = (\mu_1^\star, \mu_2^\star, \mu_3^\star)$ which he is required to tell to which message it corresponds. Since the messages m_0 and m_1 were chosen such that the label under which is created the encryption $\mu_1^\star$ (either $m_0 \| \Sigma.\mathsf{pk}$ or $m_1 \| \Sigma.\mathsf{pk}$) is different from the challenge label L, $\mathcal{M}$ can query his decryption oracle on both pairs $(\mu_1^\star, m_0 \| \Sigma.\mathsf{pk})$ or $(\mu_1^\star, m_0 \| \Sigma.\mathsf{pk})$. Result of such queries will enable $\mathcal{M}$ to open the commitment $\mu_2^\star$, and thus check the validity of the signature $\mu^\star$ w.r.t. to one of messages m_0 or m_1. Finally, when $\mathcal{R}$ outputs his answer, decryption of the challenge (c, L), $\mathcal{M}$ will simply forward this result to his challenger. □

Thus, when the considered notions are obtained from pairing a security goal GOAL $\in \{\text{OW}, \text{IND}, \text{NM}\}$ and an attack model ATK $\in \{\text{CPA}, \text{PCA}, \text{CCA}\}$, we have

Theorem 1. *The cryptosystem underlying the above construction must be at least IND-PCA secure, in case the considered reduction is key-preserving, in order to achieve INV1-CMA secure signatures.* □

Similarly to the study in [22], we generalize the above theorem to arbitrary reductions if the cryptosystem underlying the construction has a *non malleable key generator* (See Appendix C.1)

Theorem 2. *If the cryptosystem underlying the above construction has a non malleable key generator, then it must be at least IND-PCA secure in order to achieve INV1-CMA secure confirmer signatures.*

We provide the proof in Appendix C.2.

One way to explain this result is to remark that the above construction is not *strongly unforgeable*. In fact, an adversary $\mathcal{A}$, given a valid signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a message m, can create another valid signature μ' on m without the help of the signer as follows; $\mathcal{A}$ will first request the selective conversion of μ to obtain the decryption of μ_1, say r, which he will re-encrypt in μ_1' under the same label $m \| \Sigma.\mathsf{pk}$ ($\Sigma.\mathsf{pk}$ is the public key of the digital signature underlying the construction). Obviously $\mu' = (\mu_1', \mu_2, \mu_3)$ is also a valid confirmer signature on m that the signer did not produce, and thus cannot confirm/deny or convert without having access to a decryption oracle of the cryptosystem underlying the construction. This explains the insufficiency of notions like IND-CPA, and the necessity of having the cryptosystem IND-CCA secure in the invisibility claim of [35]. However, we observe that an IND-CCA secure encryption is too much than needed in this framework since a query of the type μ' is not completely uncontrolled by the signer. In fact, its first component μ_1' is an encryption of some data already disclosed by the signer, namely r, and thus a plaintext checking oracle is sufficient to deal with such a query.

Theorem 3. *The above construction is ($t, \epsilon, q_s, q_v, q_{sc}$)-INV1-CMA secure if it uses a (t, ϵ', q_s)-EUF-CMA secure digital signature, a secure commitment and a ($t + q_s q_{sc}(q_{sc} + q_v), \epsilon \cdot (1 - \epsilon')^{(q_{sc}+q_v)}, q_{sc}(q_{sc} + q_v)$)-IND-PCA secure cryptosystem with labels.*

We provide the proof in Appendix D.

4 An Efficient Construction from the "Signature of a Commitment" Paradigm

A simple way to eliminate the strong forgeability in signatures from the plain "signature of a commitment" technique consists in producing a digital signature on both the commitment and the encryption of the random string used in it. In this way, the attack discussed after Theorem 2 no longer applies, since an adversary will have to produce a digital signature on the commitment and the re-encryption of the random string used in it. We describe the full construction in the following paragraph.

4.1 Construction

Let Σ be a signature scheme given by $\Sigma.\mathsf{keygen}$ that generates $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$, $\Sigma.\mathsf{sign}$ and $\Sigma.\mathsf{verify}$. Let further Γ denote a cryptosystem given by $\Gamma.\mathsf{keygen}$ that generates $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$, $\Gamma.\mathsf{encrypt}$ and $\Gamma.\mathsf{decrypt}$. We note that Γ does need to support labels in our construction. Finally let Ω denote a commitment scheme given by $\Omega.\mathsf{commit}$ and $\Omega.\mathsf{open}$. We assume that either the ciphertexts produced by Γ or the commitment values produced by Ω do not contain a special character, say $\diamond$. The construction of confirmer signatures from Σ, Γ and Ω is given as follows.

Key generation. The signer key pair is $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$ and the confirmer key pair is $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$.

ConfirmedSign. On input message m, produce a commitment c on m using a random string r, encrypt this string in e and then produce a digital signature $\sigma = \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(e\|\diamond\|c)$. Output $\mu = (e, c, \sigma)$ as a confirmer signature on m, and prove in ZK the equality of the decryption of e and the string used for the commitment c. This proof is possible using the randomness used to encrypt r in e.

Confirmation/Denial protocol. On a message m and an alleged signature $\mu = (\mu_1, \mu_2, \mu_3)$, check the validity of μ_3 on $\mu_1\|\diamond\|\mu_2$. In case it is not valid, produce $\perp$. Otherwise, compute the decryption r of μ_1 and check whether $r \overset{?}{=} \Omega.\mathsf{open}(\mu_2, m)$, according to the result give a ZK of the equality/inequality of the decryption of c and $\Omega.\mathsf{open}(\mu_2, m)$.

Selective conversion. Proceed as in the confirmation/denial protocol with the exception of issuing the decryption of μ_1 in case the signature is valid or the symbol $\perp$ otherwise.

4.2 Security analysis

First we note that completeness, soundness and the zero knowledge property of the confirmedSign, confirmation and denial protocols are ensured by using zero knowledge proofs of knowledge. Furthermore, the construction is EUF-CMA secure and INV1-CMA secure if the underlying components are secure.

Theorem 4. *The construction depicted above is (t, ϵ, q_s)-EUF-CMA secure if uses a binding commitment scheme and a (t, ϵ, q_s)-EUF-CMA secure digital signature scheme.*

Theorem 5. *The construction depicted above is $(t, \epsilon, q_s, q_v, q_{sc})$-INV1-CMA secure if it uses a (t, ϵ', q_s)-EUF-CMA secure digital signature, a secure commitment and a $(t + q_s(q_v + q_{sc}), \frac{\epsilon}{2}(1 - \epsilon')^{q_v + q_{sc}})$-IND-CPA secure cryptosystem.*

We provide the proofs of both theorems in Appendix E.

4.3 Efficiency analysis

We show in this paragraph that requesting the cryptosystem to be only IND-CPA secure improves the efficiency of constructions from the plain "signature of a commitment" paradigm from many sides. First, it enhances the signature generation, verification and conversion cost as encryption and decryption is usually faster in IND-CPA secure encryption than in IND-CCA secure encryption (e.g., ElGamal vs Cramer-Shoup or Paillier vs Camenisch-Shoup). Next, we achieve also a shorter signature since ciphertexts produced using IND-CPA schemes are standardly shorter than their similars produced using IND-CCA secure cryptosystems. Finally, we allow homomorphic encryption in the design, which will render the confirmedSign/confirmation/denial protocols more efficient. In fact, in [16, 35], the signer/confirmer has to prove in ZK the equality/inequality of the decryption of an IND-CCA encryption and an opening value of a commitment scheme. Thus, the only efficient instantiation, that was provided, used Camenisch-Shoup encryption and Pedersen commitment. In the rest of this subsection, we enlarge the category of encryption/commitment schemes that yield efficient instantiations thanks to the allowance of homomorphic encryption in the design.

Definition 1. ***(The class $\mathbb{C}$ of commitments)*** *$\mathbb{C}$ is the set of all commitment schemes for which there exists an algorithm* Compute *that on the input: the commitment public key* pk*, the message m and the commitment c on m, computes a description of a* one-way function $f : (\mathbb{G}, *) \longrightarrow (\mathbb{H}, \circ_s)$*:*

- *where $(\mathbb{G}, *)$ is a group and $\mathbb{H}$ is a set equipped with the binary operation $\circ_s$,*
- *$\forall r, r' \in \mathbb{G}$: $f(r * r') = f(r) \circ_s f(r')$.*

and an $I \in \mathbb{H}$, such that $f(r) = I$, where r is the opening value of c w.r.t. m.

It is easy to check that Pedersen's commitment scheme is in this class. Actually, most commitment schemes have this built-in property because it is often the case that the committer wants to prove efficiently that a commitment is produced on some message. This is possible if the function f is homomorphic as shows Figure 1.

1. The prover chooses $r' \xleftarrow{R} \mathbb{G}$, computes and sends $t_1 = I \circ_s f(r')$ to the verifier.
2. The verifier chooses $b \xleftarrow{R} \{0, 1\}$ and sends it to the prover.
3. If $b = 0$, the prover sends r'. 1
   Otherwise, he sends $r * r'$.
4. If $b = 0$, the verifier checks that t_1 is computed as in Step 1.
   Otherwise, he accepts if $f(r * r') = t_1$.

Fig. 1. Proof system for membership to the language $\{r\colon f(r) = I\}$ Common input: I and Private input : r

Theorem 6. *The protocol depicted in Figure 1 is an efficient Σ protocol for proving knowledge of preimages of the function f described in Definition 1.*

The proof will be given in Appendix F.1.

For encryption, we use the same class $\mathbb{E}$ that was defined in [22], with the exception of not requiring the cryptosystems to be derived from the hybrid encryption paradigm.

Definition 2. ***(The class $\mathbb{E}$ of cryptosystems)*** *$\mathbb{E}$ is the set of encryption schemes Γ that have the following properties:*

1. *The message space is a group $\mathcal{M} = (\mathbb{G}, *)$ and the ciphertext space $\mathcal{C}$ is a set equipped with a binary operation $\circ_e$.*
2. *Let $m \in \mathcal{M}$ be a message and c its encryption with respect to a key* pk*. On the common input m and c, there exists an efficient zero knowledge proof of m being the decryption of c with respect to* pk*. The private input of the prover is either the private key* sk*, corresponding to* pk *or the randomness used to encrypt m in c.*
3. *$\forall m, m' \in \mathcal{M}$, $\forall$pk: $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m * m') = \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m) \circ_e \Gamma.\mathsf{encrypt}_{pk}(m')$. Moreover, given the randomness used to encrypt m in $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$ and m' in $\Gamma.\mathsf{encrypt}_{pk}(m')$, one can deduce (using only the public parameters) the randomness used to encrypt $m * m'$ in $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m) \circ_e \Gamma.\mathsf{encrypt}_{pk}(m')$.*

Examples of cryptosystems in the above class are ElGamal's encryption [14], the cryptosystem defined in [3] which uses the linear Diffie-Hellman KEM or Paillier's [29] cryptosystem. In fact, these cryptosystems are homomorphic and possess an efficient protocol for proving that a ciphertext decrypts to a given plaintext: the proof of equality of two discrete logarithms [9], in case of ElGamal or the cryptosystem in [3], or the proof of knowledge on an N-th root in case of Paillier's encryption.

Theorem 7. *Let Γ be a cryptosystem from the above class $\mathbb{E}$. Let furthermore e be an encryption of some message under some public* pk*. The protocol depicted in Figure 2 is an efficient Σ protocol for proving knowledge of the decryption of e.*

The proof is similar to the one given in [22]. □

```
1. The prover chooses $r' \xleftarrow{R} \mathbb{G}$, computes and sends $t_2 = \mathit{\Gamma}.\mathsf{encrypt}(r') \circ_e e$ to the verifier
2. The verifier chooses $b \xleftarrow{R} \{0,1\}$ and sends it to the signer.
3. If $b = 0$, the prover sends $r'$ and the randomness used to encrypt it in $\mathit{\Gamma}.\mathsf{encrypt}(r')$.
   Otherwise, he sends $r' * r$ and proves that $t_2$ is an encryption of $r' * r$.          1
4. If $b = 0$, the verifier checks that $t_2$ is computed as in Step 1.
   Otherwise, he checks the proof of decryption of $t_2$:
     It it fails, he rejects the proof.
```

Fig. 2. Proof system for membership to the language $\{e\colon \exists m : m = \mathit{\Gamma}.\mathsf{decrypt}(e)\}$ Common input: $(e, \mathit{\Gamma}.\mathsf{pk})$ and Private input: $\mathit{\Gamma}.\mathsf{sk}$ or randomness encrypting m in e

The confirmation/denial protocol The confirmedSign, confirmation and denial protocols of the construction in Subsection 4.1 are depicted below.

```
1. The prover and verifier, given the public input, compute $I$ as defined in Definition 1.
2. The prover chooses $r' \xleftarrow{R} \mathbb{G}$, computes and sends $t_1 = f(r') \circ_s I$ and
$t_2 = \mathit{\Gamma}.\mathsf{encrypt}(r') \circ_e e$ to the verifier.
3. The verifier chooses $b \xleftarrow{R} \{0,1\}$ and sends it to the prover.
4. If $b = 0$, the prover sends $r'$ and the randomness used to encrypt it in $\mathit{\Gamma}.\mathsf{encrypt}(r')$.
   Otherwise, he sends $r' * r$ and proves that $t_2$ is an encryption of $r' * r$.          1
5. If $b = 0$, the verifier checks that $t_1$ and $t_2$ are computed as in Step 1.
   Otherwise, he checks the proof of decryption of $t_2$:
     It it fails, he rejects the proof.
     Otherwise:
         If the prover is confirming the signature, the verifier accepts if $f(r' * r) = t_1$.
         If the prover is denying the given signature, the verifier accepts the proof if $f(r' * r) \neq t_1$.
```

Fig. 3. Proof system for membership (non membership) to the language $\{(e,c)\colon \exists r : r = \mathit{\Gamma}.\mathsf{decrypt}(e) \wedge r = (\neq)\mathit{\Omega}.\mathsf{open}(c,m)\}$ Common input: $(e, c, m, \mathit{\Gamma}.\mathsf{pk}, \mathit{\Omega}.\mathsf{pk})$ and Private input: $\mathit{\Gamma}.\mathsf{sk}$ or randomness encrypting r in e

Remark 2. The prover in Figure 3 is either the confirmer who can run the above protocols with the knowledge of his private key, or the signer who wishes to confirm the validity of a just generated signature. In fact, with the knowledge of the randomness used to encrypt s in e, the signer can issue the above confirmation protocol thanks to the properties satisfied by $\mathit{\Gamma}$.

Theorem 8. *The confirmation protocol (run either by the signer on a just generated signature or by the confirmer on any signature) described in Figure 3 is a Σ protocol.*

Theorem 9. *The denial protocol described in Figure 3 is a Σ protocol under the assumption of the underlying cryptosystem being IND-CPA-secure.*

The proofs of both theorems are given in Appendices F.2 and F.3 respectively.

5 Improvements and Possible Extensions

5.1 The “signature of an encryption” paradigm

We have seen that confirmer signatures realizing the “signature of a commitment” paradigm are comprised of a commitment on the message to be signed, an encryption of the random string used to produce the commitment, and a digital signature on the commitment. Since IND-CPA encryption can be easily used to get secure commitments, one can use instead of the commitment in the previous constructions an IND-CPA secure cryptosystem. With this choice, there will be no need of encrypting the string used to produce the encryption of the message, since the private key of the cryptosystem is sufficient to check the validity of a ciphertext w.r.t. to a given message. We give below the full description of the construction.

Key generation. The signer key pair is $(\varSigma.\mathsf{pk}, \varSigma.\mathsf{sk})$ and the confirmer key pair is $(\varGamma.\mathsf{pk}, \varGamma.\mathsf{sk})$ where $\varSigma$ and $\varGamma$ are the digital signature and the cryptosystem underlying the construction resp.

ConfirmedSign. On input message m, compute an encryption $c = \varGamma.\mathsf{encrypt}_{\varGamma.\mathsf{pk}}(m)$ of m, then a digital signature $\sigma = \varSigma.\mathsf{sign}_{\varSigma.\mathsf{sk}}(c)$. Finally output (c, σ) and a ZK proof that c decrypts in m. Such a proof is possible given the randomness used to encrypt m in c.

Confirmation/Denial protocol. On a message m and an alleged signature $\mu = (\mu_1, \mu_2)$, check the validity of μ_2 on μ_1. In case it not valid, produce $\perp$. Otherwise, compute the decryption $\tilde{m}$ of μ_1 and check whether $\tilde{m} \overset{?}{=} m$, according to the result give a ZK of the equality/inequality of the decryption of μ_1 and m. These proofs are possible using the private key of $\varGamma$.

Selective conversion. Proceed as in the confirmation/denial protocol with the exception of issuing $\perp$ is case the signature is invalid, and a *non-interactive* proof that m is the decryption of the first field of the signature otherwise.

We notice that the construction depicted above achieves better performances than all previously cited constructions in terms of signature length, generation/verification and conversion cost. In fact, the signature contains only an IND-CPA encryption and signature on it. Moreover, verification or conversion of the signature are simpler as they do not involve anymore checking whether a commitment is correctly computed. Besides, the proofs underlying the confirmedSign/confirmation/denial protocols are reduced in case of Discrete-Logarithm-based cryptosystems to proofs of equality/inequality of discrete logarithms for which there exists efficient protocols [9, 7]. The only problem with this technique is the resort to non-interactive ZK (NIZK) proofs of knowledge. In fact, we know how to produce such proofs from their interactive variants using the Fiat-Shamir paradigm, which is known to provide security only in the ROM. However, the recent results in [21, 20] exhibit efficient NIZK proofs of knowledge in some settings, which suggests that the above construction accepts efficient instantiations.

Concerning the security analysis, we first note that completeness, soundness and the ZK property of the confirmedSign/confirmation/denial protocols is ensured by the use of ZK proofs. Next, we prove that the construction is invisible and that it resists existential forgeries.

Theorem 10. *The above construction is (t, ϵ, q_s)-EUF-CMA secure if the underlying digital signature is also (t, ϵ, q_s)-EUF-CMA secure.*

Theorem 11. *The above construction is $(t, \epsilon, q_s, q_v, q_{sc})$-INV1-CMA secure if it uses a (t, ϵ', q_s)-EUF-CMA secure digital signature and a $(t + q_s(q_v + q_{sc}), \epsilon(1 - \epsilon')^{q_v + q_{sc}})$-IND-CPA secure cryptosystem.*

We provide the proofs in Appendix G.

Remark 3. Note that the IND-CPA requirement on the cryptosystem is also necessary. In fact, deterministic schemes, e.g., RSA (which is OW-CPA secure) are not allowed in the design, since an invisibility adversary will compute the encryptions of the two challenge messages and check whether one of them is the first field of the signature.

5.2 A stronger security model

In [11], the author presented an elaborate security model. We discuss in this paragraph how one can extend the constructions seen so far to this model.

Security against malicious confirmers. The first difference between our model and the one in [11] is the unforgeability against *malicious* confirmers which is satisfied in the latter but not considered in the former. This property requires the construction to remain EUF-CMA secure even if the EUF-CMA adversary is allowed to choose the confirmer public key. One can easily see that the constructions presented in this document meet this property as the confirmer public key does not play any role in the unforgeability proofs.

Correctness of the conversion. Another difference lies in requesting the confirmer to provide a proof of the correctness of the conversion. This is vital, because in all constructions that realize the "signature of a commitment" paradigm, the confirmer can convert *invalid* signatures; he can release the "real" opening value of the commitment, which does not have to be the decryption of the first field of the confirmer signature. A way to overcome this, is to provide, along with the opening value of the commitment, a proof that it is the correct decryption of the first field of the alleged signature. In [11], the author suggested to use a protocol, i.e., an interactive proof, proving the correctness of the conversion. We propose to use non interactive proofs to get transferability, i.e., anybody can check the correctness of a converted confirmer signature. As mentioned in the previous subsection, there exists efficient ways to obtain non-interactive proofs of knowledge without using Fiat-Shamir heuristics. Again constructions shown before meet this stronger property (the reduction in the invisibility proofs can issue such proofs using the randomness used to produce the encryption of the commitment opening value). Finally, our constructions allow also the confirmer to convert *invalid* signatures although it is not his responsibility to convert ill-formed signatures. The confirmer can do so by issuing simply the decryption of the first field of signature (in case it is a well-formed ciphertext) along with a non-interactive proof of the correctness of the decryption. Anybody can then check that the released string does not open the commitment (second field of the confirmer signature).

6 Summary

We supplemented the study in [22]. In fact, after a quick browse through the plethora of generic constructions of confirmer signatures, we managed to categorize them under either those instantiating the "encryption of a signature" principle, or those realizing the "signature of commitment" paradigm. Constructions obtained from both *plain* paradigms were shown to necessitate strong encryption which makes them quite impractical, or at least allow very limited instantiations. However, a small variation of both principles results in a tremendous improvement: short signature, small generation, verification and conversion cost, in addition to efficient confirmation/denial protocols. The "encryption of a signature" principle compares better than the "signature of commitment" paradigm in terms of security (potential anonymity) and length of the resulting signatures, however, the latter betters the former in terms of flexibility as it applies to *any* signature. We also shed light on a particular construction, which can be seen as a special sub-case of the latter paradigm, namely the "signature of an encryption" technique. The advantage of this technique consists in achieving better performances than the original technique (short signature, small generation, verification and conversion cost) , yet applying to any signature scheme. Its sole limitation resides in requiring efficient non interactive proofs of knowledge. This motivates research to further tackle this problem as was started recently in [21].

References

1. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, *Relations Among Notions of Security for Public-Key Encryption Schemes.*, Advances in Cryptology - CRYPTO'98 (H. Krawczyk, ed.), LNCS, vol. 1462, Springer, 1998, pp. 26–45.
2. M. Bellare and P. Rogaway, *The Exact Security of Digital Signatures: How to Sign with RSA and Rabin.*, in Maurer [24], pp. 399–416.
3. D. Boneh, X. Boyen, and H. Shacham, *Short Group Signatures.*, Advances in Cryptology - CRYPTO 2004 (M. K. Franklin, ed.), LNCS, vol. 3152, Springer, 2004, pp. 41–55.
4. D. Boneh and R. Venkatesan, *Breaking RSA May Not Be Equivalent to Factoring.*, in Nyberg [27], pp. 59–71.
5. G. Brassard, D. Chaum, and C. Crépeau, *Minimum disclosure proofs of knowledge.*, J. Comput. Syst. Sci. **37** (1988), no. 2, 156–189.
6. J. Camenisch and M. Michels, *Confirmer Signature Schemes Secure against Adaptative Adversaries.*, Advances in Cryptology - EUROCRYPT 2000 (B. Preneel, ed.), LNCS, vol. 1807, Springer, 2000, pp. 243–258.
7. J. Camenisch and V. Shoup, *Practical Verifiable Encryption and Decryption of Discrete Logarithms.*, Advances in Cryptology - CRYPTO 2003 (D. Boneh, ed.), LNCS, vol. 2729, Springer, 2003, pp. 126–144.
8. D. Chaum, *Designated Confirmer Signatures.*, Advances in Cryptology - EUROCRYPT'94 (A. De Santis, ed.), LNCS, vol. 950, Springer, 1995, pp. 86–91.
9. D. Chaum and T. P. Pedersen, *Wallet Databases with Observers.*, Advances in Cryptology - CRYPTO'92 (E. F. Brickell, ed.), LNCS, vol. 740, Springer, 1993, pp. 89–105.
10. D. Chaum and H. van Antwerpen, *Undeniable Signatures.*, Advances in Cryptology - CRYPTO'89 (G. Brassard, ed.), LNCS, vol. 435, Springer, 1990, pp. 212–216.
11. D. Wikström, *Designated Confirmer Signatures Revisited*, TCC 2007 (S. P. Vadhan, ed.), L, vol. 4392, Springer, 2007, pp. 342–361.
12. I. B. Damgård and T. P. Pedersen, *New Convertible Undeniable Signature Schemes.*, in Maurer [24], pp. 372–386.
13. W. Diffie and M. E. Hellman, *New Directions in Cryptography.*, IEEE Trans. Inf. Theory **22** (1976), 644–654.
14. T. El Gamal, *A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms.*, IEEE Trans. Inf. Theory **31** (1985), 469–472.
15. S. D. Galbraith and W. Mao, *Invisibility and Anonymity of Undeniable and Confirmer Signatures.*, Topics in Cryptology - CT-RSA 2003 (M. Joye, ed.), LNCS, vol. 2612, Springer, 2003, pp. 80–97.
16. C. Gentry, D. Molnar, and Z. Ramzan, *Efficient Designated Confirmer Signatures Without Random Oracles or General Zero-Knowledge Proofs*, in Roy [33], pp. 662–681.
17. O. Goldreich, *Foundations of cryptography. Basic Tools.*, Cambridge University Press., 2001.
18. S. Goldwasser, S. Micali, and R. L. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.*, SIAM J. Comput. **17** (1988), no. 2, 281–308.
19. S. Goldwasser and E. Waisbard, *Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes.*, Theory of Cryptography, TCC 2004 (M. Naor, ed.), LNCS, vol. 2951, Springer, 2004, pp. 77–100.
20. J. Camenisch and N. Chandran and V. Shoup, *A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks*, EUROCRYPT 2009 (A. Joux, ed.), LNCS, vol. 5479, Springer, 2009, pp. 351–368.
21. J. Groth and A. Sahai, *Efficient Non-interactive Proof Systems for Bilinear Groups*, EUROCRYPT 2008 (N. P. Smart, ed.), LNCS, vol. 4965, Springer, 2008, pp. 415–432.
22. L. El Aimani, *On Generic Constructions of Designated Confirmer Signatures (The "Encryption of a Signature" Paradigm Revisited)*, Cryptology ePrint Archive, Report 2009/403, 2009, `http://eprint.iacr.org/`.
23. C. H. Lim and P. J. Lee, *Modified Maurer-Yacobi's scheme and its applications.*, Advances in Cryptology - AUSCRYPT '92 (J. Seberry and Y. Zheng, eds.), LNCS, vol. 718, Springer, 1993, pp. 308–323.
24. U. M. Maurer (ed.), *Advances in Cryptology* - EUROCRYPT*'96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, LNCS, vol. 1070, Springer, 1996.
25. M. Michels and M. Stadler, *Generic Constructions for Secure and Efficient Confirmer Signature Schemes.*, in Nyberg [27], pp. 406–421.
26. J. Monnerat and S. Vaudenay, *Chaum's Designated Confirmer Signature Revisited.*, Information Security, ISC 2005 (J. Zhou, J. Lopez, R. H. Deng, and F. Bao, eds.), LNCS, vol. 3650, Springer, 2005, pp. 164–178.
27. K. Nyberg (ed.), *Advances in Cryptology* - EUROCRYPT*'98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, LNCS, vol. 1403, Springer, 1998.
28. T. Okamoto, *Designated Confirmer Signatures and Public-Key Encryption are Equivalent.*, Advances in Cryptology - CRYPTO'94 (Y. Desmedt, ed.), LNCS, vol. 839, Springer, 1994, pp. 61–74.
29. P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, EUROCRYPT (J. Stern, ed.), LNCS, vol. 1592, Springer, 1999, pp. 223–238.
30. ______, *Impossibility Proofs for RSA Signatures in the Standard Model*, CT-RSA (M. Abe, ed.), LNCS, vol. 4377, Springer, 2007, pp. 31–48.
31. P. Paillier and J. Villar, *Trading One-Wayness Against Chosen-Ciphertext Security in Factoring-Based Encryption*, ASIACRYPT (X. Lai and K. Chen, eds.), LNCS, vol. 4284, Springer, 2006, pp. 252–266.

32. P. Paillier and D. Vergnaud, *Discrete-Log Based Signatures May Not Be Equivalent to Discrete-Log.*, in Roy [33], pp. 1–20.
33. B. Roy (ed.), *Advances in Cryptology* - ASIACRYPT *2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Taj Coromandel, Chennai, India December 4-8, 2005, Proceedings*, LNCS, vol. 3788, Springer, 2005.
34. V. Shoup and R. Gennaro, *Securing Threshold Cryptosystems against Chosen Ciphertext Attack*, J. Cryptology **15** (2002), no. 2, 75–96.
35. G. Wang, J Baek, D. S. Wong, and F. Bao, *On the Generic and Efficient Constructions of Secure Designated Confirmer Signatures*, PKC 2007 (T. Okamoto and X. Wang, eds.), LNCS, vol. 4450, Springer, 2007, pp. 43–60.

A Preliminaries

A.1 Comparison with other CDCS security models

First we note that requiring the confirmedSign, confirmation and denial protocols to be zero knowledge captures the different definitions of non transferability which were provided in [6, 19, 16, 35, 11]. Moreover, combination of this requirement and the invisibility property implies the transcript simulatability defined in [16, 35]. Finally, in some applications, it is required that the confirmer signatures are anonymous, i.e., do not leak the identity (public key) of the signer (see [15]). Thus, to capture both anonymity and invisibility, Galbraith and Mao introduced in [15] a notion, which we denote INV2-CMA, that requires the confirmer signatures to be indistinguishable from random elements in the signature space. This new notion is proven to imply both INV1-CMA and ANO-CMA (Theorem 1 and Theorem 4 respectively of [15]). The constructions analyzed/given in this paper, for instance those described in [16, 35] do not fulfill this notion. However, this should not be a problem because the INV1-CMA property suffices in many practical situations. We refer to the discussion in [16] (Section 3) for techniques that can be used by the signer to camouflage the presence of valid signatures.

A.2 Digital signatures

A signature scheme Σ comprises three algorithms, namely the key generation algorithm keygen, the signing algorithm sign, and the verification algorithm verify. The standard security notion for a signature scheme is existential unforgeability under chosen message attacks (EUF-CMA), which was introduced in [18]. Informally, this notion refers to the hardness of, given a signing oracle, producing a valid pair of message and corresponding signature such that message has not been queried to the signing oracle. There exists also the stronger notion, SEUF-CMA (strong existential unforgeability under chosen message attack), which allows the adversary to produce a forgery on a previously queried message, however the corresponding signature must not be obtained from the signing oracle.

A.3 Public key encryption schemes

A public key encryption (PKE) scheme consists of the key generation algorithm keygen, the encryption algorithm encrypt and the decryption algorithm decrypt. The typical *security goals* a cryptosystem should attain are: one-wayness (OW) which corresponds to the difficulty of recovering the plaintext from a ciphertext, indistinguishability (IND) which refers to the hardness of distinguishing ciphertexts based on the messages they encrypt, and finally non-Malleability (NM) which corresponds to the hardness of deriving from a given ciphertext another ciphertext such that the underlying plaintexts are meaningfully related. Conversely, the typical *attack models* an adversary against an encryption scheme is allowed to are: Chosen Plaintext Attack (CPA) where the adversary can encrypt any message of his choice. This is inevitable in public key settings, Plaintext Checking Attack (PCA) in which the adversary is allowed to query an oracle on pairs (m, c) and gets answers whether m is really encrypted in c or not, and finally Chosen Ciphertext Attack (CCA) where the adversary is allowed to query a decryption oracle. Pairing the mentioned goals with these attack models yields nine *security notions*: GOAL-ATK for GOAL $\in \{\text{OW}, \text{IND}, \text{NM}\}$ and ATK $\in \{\text{CPA}, \text{PCA}, \text{CCA}\}$. We refer to [1] for the formal definitions of these notions as well as for the relations they satisfy.

Cryptosystems with labels. Encryption with labels was first introduced in [34]. In these schemes, the encryption algorithm takes as input, in addition to the public key pk and the message m intended to be encrypted, a label L. Similarly, the decryption algorithm takes additionally to the ciphertext and private key the label under which the ciphertext was created. Security notions are then defined as usual except that the adversary specifies always the label, to be used in the challenge ciphertext, to his challenger, and in case he (the adversary) is allowed to query oracles, then he cannot query them on the pair formed by the challenge and the label used to form it.

A.4 Commitment schemes

A commitment scheme [5] consists of the following algorithms:

- setup: the setup algorithm that generates the public parameters of the system.
- keygen: generates probabilistically a public commitment key pk.
- commit: a probabilistic algorithm that, on input a public key pk and a message m, produces a pair (c, r): c serves as the commitment value (locked box), and r as the opening value.
- open: this is a deterministic algorithm that given a commitment (c, r), w.r.t. a public key pk, on a alleged message m, checks whether $c \overset{?}{=} \mathsf{commit}_{\mathsf{pk}}(m, r)$.

The algorithm open must succeed if the commitment was correctly formed (correctness). Moreover, we require the following security properties:

1. **Hiding.** It is hard for an adversary A to generate two messages m_0, m_1 such that he can distinguish between their corresponding locked boxes c_o, c_1. That is, c reveals no information about m.
2. **Binding.** It is hard for an adversary A to come up with a *collision* (c, d, d') such that (c, d) and (c, d') are valid commitments for m and m' resp and $m \neq m'$.

We call a commitment scheme *secure* if it meets the previous properties.

It is easy to see the similarity between public key encryption and commitment schemes. In fact, one can easily check that IND-CPA encryption implies a secure commitment scheme. The main difference between encryption and commitment is that the former requires the decryption algorithm to be based on a "universal" secret key (independent of the message) whilst commitment allows to decrypt with a "message-dependent" secret key, namely the opening value r of the message in question. Another difference is that in encryption, the message is always derived from the ciphertext. This is not always the case in commitments, as the following example shows:

- setup and keygen choose a multiplicative group $(\mathbb{G}, \cdot)$ of order d and generated by an element g. Choose further an element $y \in \mathbb{G}$ of unknown discrete logarithm with respect to g, and a collision resistant hash function $h\colon \{0,1\}^\star \to \mathbb{Z}_d$. The public commitment key is y.
- commit on a message $m \in \{0,1\}^\star$ is the pair (r, c) where $r \xleftarrow{R} \mathbb{Z}_d$ and $c = g^r y^{h(m)}$.
- open an alleged commitment (c, r) on a message m is achieved by checking whether $c \overset{?}{=} g^r y^{h(m)}$.

It is easy to check that the above commitment, referred to as *Pedersen-based* commitment scheme, is correct. Moreover it is statistically hiding because r is random in $\mathbb{Z}_d$ and so is $c = g^r y^{h(m)}$, regardless of m. Besides the biding property is achieved under the discrete logarithm assumption in $\mathbb{G}$ and the collision resistance assumption on the hash function h.

Finally, it is worth noting that given an alleged commitment value c on a message m, one can use the opening value r to prove (disprove) in zero knowledge that c is (is not) a commitment on m. In fact, the last assertion corresponds to an NP (co-NP) language which accepts a zero knowledge proof system (see [17]).

A.5 Σ protocols

A Σ protocol is an argument of knowledge which is complete, sound, Zero Knowledge (ZK), and which remains ZK after parallel repetition. We refer to [17] for more information.

B Security of the plain "Signature of a Commitment" Paradigm

Theorem 12. *The construction depicted in Section 3 is (t, ϵ, q_s)-EUF-CMA secure if uses a binding commitment scheme and a (t, ϵ, q_s)-EUF-CMA secure digital signature scheme.*

Proof. Let $\mathcal{A}$ be an attacker against the construction. We will construct an attacker $\mathcal{R}$ against the underlying signature scheme as follows.

$\mathcal{R}$ gets the parameters of the signature scheme Σ from his challenger, namely the public key $\Sigma.\mathsf{pk}$. Then, $\mathcal{R}$ will choose an appropriate cryptosystem Γ along with the key pair $(\Gamma.\mathsf{sk}, \Gamma.\mathsf{pk})$ and a suitable commitment scheme Ω. Finally, $\mathcal{R}$ will set the mentioned entities as components of the construction $\mathcal{A}$ is trying to attack.

For a signature query on a message m_i, $\mathcal{R}$ will first create a commitment c_i using a random string r_i, then he will query his own challenger for a digital signature on c_i. Let σ_i be the output digital signature on c_i. The output confirmer signature consists of the triple $\mu_i = (e_i, c_i, \sigma_i)$, where e_i is an encryption of r_i under the label $m_i \| \Sigma.\mathsf{pk}$.

$\mathcal{A}$ will have at his disposal $\Gamma.\mathsf{sk}$ and thus he won't need to ask confirm/deny or selective conversion queries. And, even in case he requests them, $\mathcal{R}$ is able to answer such queries with the knowledge of $\Gamma.\mathsf{sk}$.

At some point, $\mathcal{A}$ will output a forgery $\mu^\star = (e^\star, c^\star, \sigma^\star)$ on some message $m^\star$ that has never been queried. If there exists an $1 \leq i \leq q_s$ such that $c^\star = c_i$, where $\mu_i = (e_i, c_i, \sigma_i)$ is an output confirmer signature on a query m_i, then since $m_i \neq m^\star$, $\mathcal{R}$ will output a collision for the commitment scheme Ω. Since the latter is by assumption binding, $c^\star$ never occurred in signatures output to $\mathcal{A}$. Therefore $(c^\star, \sigma^\star)$ corresponds to a valid existential forgery on Σ. □

Theorem 13. *The construction depicted in Section 3 is $(t, \epsilon, q_s, q_v, q_{sc})$-INV1-CMA secure if is (t, ϵ', q_s)-EUF-CMA secure and it is uses a hiding commitment and a $(t, \frac{\epsilon}{2}(1 - \epsilon')^{q_v + q_{sc}}, q_s)$-IND-CCA secure cryptosystem with labels.*

Proof. Let $\mathcal{A}$ be an attacker against the construction. We will construct an attacker $\mathcal{R}$ against the underlying cryptosystem scheme as follows.

$\mathcal{R}$ gets the parameters of the cryptosystem Γ from his challenger. Then he will choose a signature scheme Σ (along with a key pair $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$) and a secure commitment scheme Ω. $\mathcal{R}$ will set the above entities as components of the construction $\mathcal{A}$ is trying to attack.

For a signature query on a message m_i, $\mathcal{R}$ will compute a commitment c_i on m_i using a random string r_i, which he will encrypt in e_i under the label $m_i \| \Sigma.\mathsf{pk}$, then he will produce a digital signature σ_i on c_i using $\Sigma.\mathsf{sk}$. Finally he outputs $\mu_i = (e_i, c_i, \sigma_i)$ as a confirmer signature on m_i and a ZK proof of knowledge of the equality of the decryption of e_i and the string used in the commitment c_i. Such a proof is possible using the randomness t_i used to encrypt r_i in e_i..

To confirm/deny an alleged signature $\mu_i = (\mu_i^1, \mu_i^2, \mu_i^3)$ on a message m_i, $\mathcal{R}$ will proceed as follows. First he checks the validity of the digital signature μ_i^3 on μ_i^2, in case it is invalid, he will output $\perp$, otherwise he will obtain the decryption of μ_i^1 (from the decryption oracle thanks to the CCA attack model), r_i; if r_i is (is not) the same string used to compute the commitment μ_i^2, $\mathcal{R}$ will issue a zero knowledge proof of the equality (inequality) of the decryption of μ_i^1 and the string used for the commitment μ_i^2. $\mathcal{R}$ can issue these proofs without the knowledge of $\Gamma.\mathsf{sk}$ using the rewinding technique (the proofs are ZK and thus simulatable) or by keeping a record of the randomnesses used to encrypt the random strings r_i

in e_i. Selective conversion is similarly carried out with the exception of issuing the decryption of μ_i^1 in case the signature is valid and $\perp$ otherwise.

At some point, $\mathcal{A}$ will output two messages m_0, m_1 that have not been queried for signature. $\mathcal{R}$ will then choose uniformly at random a bit $b \overset{R}{\longleftarrow} \{0,1\}$, and two different random strings r_0 and r_1 from the corresponding space. $\mathcal{R}$ will output to his challenger the label $m_b \| \Sigma.\mathsf{pk}$ and the strings r_0, r_1. He receives then a ciphertext c, encryption of $r_{b'}$, for some $b' \overset{R}{\longleftarrow} \{0,1\}$. To answer his challenger, $\mathcal{R}$ will compute a commitment c_b on the message m_b using the string $r_{b''}$ where $b'' \overset{R}{\longleftarrow} \{0,1\}$. Then, $\mathcal{R}$ will output $\mu = (c, c_b, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c_b))$ as a challenge signature to $\mathcal{A}$. Note that $\mathcal{A}$ can only exploit information leaked from c about the opening value of c_b because the commitment scheme is by assumption hiding.

Next, $\mathcal{A}$ will continue issuing queries which $\mathcal{R}$ can handle as previously, with the exception of issuing the denial protocol in case of a verification query (or $\perp$ in case of a selective conversion query) on a presumed signature $(c, -, -)$ on m_b. In fact, in this phase, $\mathcal{R}$ cannot query his decryption oracle on $(c, m_b \| \Sigma.\mathsf{pk})$. This simulation differs from the real algorithm when the signature $(c, -, -)$ is valid. Since $\mathcal{A}$ is not allowed to query m_0, m_1 to the signing oracle nor (μ, m_i) ($i \in \{0,1\}$) to the verification oracle, such a query will correspond to an existential forgery on the construction as m_b was never queried to the signing oracle. Thus, the probability that this does not occur is at least $(1-\epsilon')^{q_v+q_{sc}}$ since the construction is (t, ϵ', q_s)-EUF-CMA secure by assumption.

Now, let us analyze the challenge signature $\mu = (c, c_b, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c_b))$. In case, c is an encryption of $r_{b''}$ (that is if $b' = b''$), then μ corresponds to a valid confirmer signature on m_b. Otherwise, it is not a valid signature on neither m_b nor m_{1-b}. In fact, c_b is a commitment on m_b using a string different from the decryption of c under the label $m_b \| \Sigma.\mathsf{pk}$. Let b_a the bit output by $\mathcal{A}$. $\mathcal{A}$ will output b'' to his challenger in case $b = b_a$ and $1 - b''$ otherwise.

The advantage of $\mathcal{A}$ in such an attack is defined by

$$\epsilon = \mathsf{adv}(\mathcal{A}) = \Pr[b_a = b | b' = b''] - \frac{1}{2}$$

Whereas the advantage of $\mathcal{R}$ is given by

$$\begin{aligned}
\mathsf{adv}(\mathcal{R}) &= (1-\epsilon')^{q_v+q_{sc}} \left[\Pr[b = b_a, b' = b''] + \Pr[b \neq b_a, b' \neq b''] - \frac{1}{2}\right] \\
&= (1-\epsilon')^{q_v+q_{sc}} \left[\Pr[b = b_a | b' = b''] \Pr[b' = b''] + \Pr[b \neq b_a | b' \neq b''] \Pr[b' \neq b''] - \frac{1}{2}\right] \\
&= (1-\epsilon')^{q_v+q_{sc}} \left[\frac{1}{2}(\epsilon + \frac{1}{2}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2}\right] \\
&= \frac{\epsilon}{2}(1-\epsilon')^{q_v+q_{sc}}
\end{aligned}$$

The last but one equation is due to the facts $\Pr[b' \neq b''] = \Pr[b' = b''] = \frac{1}{2}$ as $b'' \overset{R}{\longleftarrow} \{0,1\}$, and to the fact that, in case $b' \neq b''$, the probability that $\mathcal{A}$ answers b is exactly $\frac{1}{2}$ since in that case the challenge signature is not valid on both messages. □

C Generalization to Arbitrary Reductions

C.1 Non malleable key generators

We define the notion of *non malleability of a cryptosystem key generator* through the following two games:

In **Game 0**, we consider an algorithm $\mathcal{R}$ trying to break a cryptosystem Γ , w.r.t. a public key $\Gamma.\mathsf{pk}$, in the sense of NM-CPA (or OW-CCA) using an adversary $\mathcal{A}$ which solves a problem A, perfectly reducible to

OW-CPA breaking the cryptosystem Γ (w.r.t. the public key $\Gamma.\mathsf{pk}$). In this game, $\mathcal{R}$ lunches $\mathcal{A}$ over his own challenge key $\Gamma.\mathsf{pk}$ and some other parameters chosen freely by $\mathcal{R}$. We will denote by $\mathsf{adv}_0(\mathcal{R}^{\mathcal{A}})$ the success probability of $\mathcal{R}$ in such a game, where the probability is taken over the random tapes of both $\mathcal{R}$ and $\mathcal{A}$. We further define $\mathsf{succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A}) = \max_{\mathcal{R}} \mathsf{adv}_0(\mathcal{R}^{\mathcal{A}})$ to be the success in **Game 0** of the best reduction $\mathcal{R}$ making the best possible use of the adversary $\mathcal{A}$. Note that the goal of **Game 0** is to include all key-preserving reductions $\mathcal{R}$ from NM-CPA (or OW-CCA) breaking the cryptosystem in question to solving a problem A, which is reducible to OW-CPA breaking the same cryptosystem.
In **Game 1**, we consider the same entities as in **Game 0**, with the exception of providing $\mathcal{R}$ with, in addition to $\mathcal{A}$, a OW-CPA oracle (i.e. a decryption oracle corresponding to Γ) that he can query w.r.t. any public key $\Gamma.\mathsf{pk}' \neq \Gamma.\mathsf{pk}$, where $\Gamma.\mathsf{pk}$ is the challenge public key of $\mathcal{R}$. Similarly, we define $\mathsf{adv}_1(\mathcal{R}^{\mathcal{A}})$ to be the success of $\mathcal{R}$ in such a game, and $\mathsf{succ}_{\Gamma}^{\mathsf{Game1}}(\mathcal{A}) = \max_{\mathcal{R}} \mathsf{adv}_0(\mathcal{R}^{\mathcal{A}})$ the success in **Game 1** of the reduction $\mathcal{R}$ making the best possible use of the adversary $\mathcal{A}$ and of the decryption (OW-CPA) oracle.

Definition 3. *A cryptosystem Γ is said to have a non malleable key generator if $\Delta = max_{\mathcal{A}} |\mathsf{succ}_{\Gamma}^{\mathsf{Game1}}(\mathcal{A}) - \mathsf{succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A})|$ is negligeable in the security parameter.*

This definition informally means that a cryptosystem has a non malleable key generator if NM-CPA (or OW-CCA) breaking it w.r.t. a key pk is no easier when given access to a decryption (OW-CPA) oracle w.r.t. any public key $\mathsf{pk}' \neq \mathsf{pk}$.

C.2 Proof of Theorem 2

To prove Theorem 2, we first need the following Lemma (similar to Lemma 6 of [31])

Lemma 3. *Let $\mathcal{A}$ be an adversary solving a problem A, reducible to OW-CPA breaking a cryptosystem Γ, and let $\mathcal{R}$ be an arbitrary reduction $\mathcal{R}$ that NM-CPA (OW-CCA) breaks a cryptosystem Γ, given access to $\mathcal{A}$. We have*

$$\mathsf{adv}(\mathcal{R}) \leq \mathsf{succ}_{\Gamma}^{\mathsf{Game1}}(\mathcal{A})$$

Proof. We will construct an algorithm $\mathcal{M}$ that plays **Game 1** with respect to a perfect oracle for $\mathcal{A}$ and succeeds in breaking the NM-CPA (OW-CCA) security of Γ with the same success probability of $\mathcal{R}$. Algorithm $\mathcal{M}$ gets a challenge w.r.t. a public key pk and launches $\mathcal{R}$ over the same challenge and the same public key. If $\mathcal{R}$ calls $\mathcal{A}$ on pk, then $\mathcal{M}$ will call his own oracle for $\mathcal{A}$. Otherwise, if $\mathcal{R}$ calls $\mathcal{A}$ on $\mathsf{pk}' \neq \mathsf{pk}$, $\mathcal{M}$ will invoke his own decryption oracle for pk' (OW-CPA oracle) to answer the queries. In fact, by assumption, the problem A is reducible to OW-CPA solving Γ. Finally, when $\mathcal{R}$ outputs the result to $\mathcal{M}$, the latter will output the same result to his own challenger. □

Proof of Theorem 2 This proof is similar to the one of Theorem 5 in [31].

Proof. We first remark that the invisibility of the construction depicted in Section 3 is perfectly reducible to OW-CPA breaking the cryptosystem underlying the construction. In fact, an invisibility adversary $\mathcal{A}$, given a challenge confirmer signature can first decrypt its first component, then use the resulting string to check the validity of the second component (alleged commitment on the message in question).
Next, we note that the advantage of the meta-reduction $\mathcal{M}$ in the proof of Lemma 1 (Lemma 2) is the same as the advantage of any key-preserving reduction $\mathcal{R}$ reducing the invisibility of a given confirmer signature to the NM-CPA (OW-CCA) security of its underlying cryptosystem Γ. For instance, this applies to the reduction making the best use of an invisibility adversary $\mathcal{A}$ against the construction. Therefore we have:

$$\mathsf{succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A}) \leq \mathsf{succ}(NM - CPA[\Gamma])$$

where $\mathsf{succ}(NM - CPA[\mathit{\Gamma}])$ is the success of breaking $\mathit{\Gamma}$ in the NP-CPA sense. We also have

$$\mathsf{succ}_{\mathit{\Gamma}}^{\mathsf{Game0}}(\mathcal{A}) \leq \mathsf{succ}(OW - CCA[\mathit{\Gamma}])$$

.
Now, Let $\mathcal{R}$ be an arbitrary reduction from NM-CPA (OW-CCA) breaking a cryptosystem $\mathit{\Gamma}$, with a non malleable key generator, to INV1-CMA breaking the construction (using the same cryptosystem $\mathit{\Gamma}$). We have

$$\begin{aligned} \mathsf{adv}(\mathcal{R}) &\leq \mathsf{succ}_{\mathit{\Gamma}}^{\mathsf{Game1}}(\mathcal{A}) \\ &\leq \mathsf{succ}_{\mathit{\Gamma}}^{\mathsf{Game0}}(\mathcal{A}) + \mathit{\Delta} \\ &\leq \mathsf{succ}(NM - CPA[\mathit{\Gamma}])(\mathsf{succ}(OW - CCA[\mathit{\Gamma}])) + \mathit{\Delta} \end{aligned}$$

since $\mathit{\Delta}$ is negligeable, then under the assumption of $\mathit{\Gamma}$ being NM-CPA (OW-CCA) secure, the advantage of $\mathcal{R}$ is also negligeable. □

D Proof of Theorem 3

Proof. Let $\mathcal{A}$ be an attacker against the construction. We will construct an attacker $\mathcal{R}$ against the underlying cryptosystem scheme as follows.

$\mathcal{R}$ gets the parameters of the cryptosystem $\mathit{\Gamma}$ from his challenger. Then he will choose a signature scheme $\mathit{\Sigma}$ (along with a key pair $(\mathit{\Sigma}.\mathsf{pk}, \mathit{\Sigma}.\mathsf{sk})$) and a suitable commitment scheme $\mathit{\Omega}$. $\mathcal{R}$ will set the above entities as components of the construction $\mathcal{A}$ is trying to attack.

For a signature query on a message m_i, $\mathcal{R}$ will compute a commitment c_i on m_i using a random string r_i, which he will encrypt in e_i under the label $m_i \| \mathit{\Sigma}.\mathsf{pk}$, then he will produce a digital signature σ_i on c_i using $\mathit{\Sigma}.\mathsf{sk}$. Next, he outputs $\mu_i = (e_i, c_i, \sigma_i)$ as a confirmer signature on m_i and a ZK proof of knowledge of the equality of the decryption of e_i and the string used in the commitment c_i. Such a proof is possible using the randomness t_i used to encrypt r_i in e_i. Finally, $\mathcal{R}$ will add the record $R_i = (m_i, t_i, r_i, e_i, c_i, \sigma_i)$ to a history list $\mathcal{L}$.

To confirm/deny an alleged signature $\mu_i = (\mu_i^1, \mu_i^2, \mu_i^3)$ on a message m_i, $\mathcal{R}$ will proceed as follows. First he checks the validity of the digital signature μ_i^3 on μ_i^2, in case it is invalid, he will output $\perp$, otherwise he will check the list $\mathcal{L}$, if he finds a record R_i having as first field the message m_i, he will proceed to the next step, namely, check whether the fourth field of R_i is equal to μ_i^1, if it is the case, $\mathcal{R}$ will issue a ZK proof of the equality of the decryption of μ_i^1 and the string used for the commitment μ_i^2. $\mathcal{R}$ can issue these proofs without the knowledge of $\mathit{\Gamma}.\mathsf{sk}$ using the rewinding technique (the proofs are ZK and thus simulatable) or by using the second field of R_i (randomness used to produce the encryption μ_i^1). Now, if R_i contains m_i in its first field, but its fourth field is different from μ_i^1, then $\mathcal{R}$ will check the next record R_j ($j > i$) having m_i in its first field and proceed in a similar fashion. Actually, if the message m_i is queried more than once, then it will occur in many records in $\mathcal{L}$. If $\mathcal{R}$ browses through all the records but none of them contains m_i and μ_i^1 in their first and fourth field resp, then for all the records R_i containing m_i in their first field, $\mathcal{A}$ will invoke his PCA oracle on the ciphertext μ_i^1 and the third fields of these records. If one of the queries yields "yes" as an answer, e.g., there exists a record $R_j = (m_i, t_j, r_j, e_j, c_j, \sigma_j)$ such that its third field r_j is a decryption of μ_i^1, then according to whether r_j is (is not) the opening value of the commitment μ_i^2 on m_i, $\mathcal{R}$ will issue a ZK proof of the equality (inequality) of the decryption of μ_i^1 and the string used for the commitment μ_i^2. Again such a proof is possible to issue using the rewinding technique (the value t_j cannot be used here because it was not used to encrypt r_j in μ_i^1). Finally, if no query to the PCA oracle yields the answer "yes", then $\mathcal{R}$ will issue the denial protocol, namely simulate a ZK proof, using the rewinding technique, of the inequality of the decryption of μ_i^1 and of the string used for the commitment μ_i^2.

Selective conversion is similarly carried out with the exception of issuing the decryption of μ_i^1 instead of the confirmation protocol and $\perp$ instead of the denial protocol.

The difference between the above simulation and the real execution of the algorithm is when the signature $\mu_i = (\mu_i^1, \mu_i^2, \mu_i^3)$ is valid, however, μ_i^1 is not an encryption of a string r_i already issued to $\mathcal{A}$ during a selective conversion query regarding the message m_i and a presumed signature on it. We distinguish two cases, either m_i was never queried for signature, in which case such a signature would correspond to an existential forgery on the construction and thus to an existential forgery on the underlying digital signature. Or, m_i was queried before for signature. Let $\mu_j = (\mu_j^1, \mu_j^2, \mu_j^3)$ be the output confirmer signature to such a query. Since μ_i^1 is encryption of some r_i which was never used to generate signatures on m_i, then with overwhelming probability $\mu_i^2 \neq \mu_j^2$ (both are commitment on m_i with different random strings). Thus, in this case (μ_i^2, μ_i^3) will correspond to an existential forgery on the underlying digital signature scheme. We conclude that the above simulation is indistinguishable from the real execution with probability at least $(1-\epsilon')^{q_v+q_{sc}}$, as the digital signature scheme underlying the construction is (t, ϵ', q_s)-EUF-CMA secure by assumption.

At some point, $\mathcal{A}$ will output two messages m_0, m_1 that have not been queried for signature. The latter will then choose uniformly at random a bit $b \xleftarrow{R} \{0,1\}$, and two different random strings r_0 and r_1 from the corresponding space. $\mathcal{R}$ will output to his challenger the label $m_b \| \Sigma.\mathsf{pk}$ and the strings r_0, r_1. He receives then a ciphertext c, encryption of $r_{b'}$, for some $b' \xleftarrow{R} \{0,1\}$. To answer his challenger, $\mathcal{R}$ will compute a commitment c_b on the message m_b using the string $r_{b''}$ where $b'' \xleftarrow{R} \{0,1\}$. Then, $\mathcal{R}$ will output $\mu = (c, c_b, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c_b))$ as a challenge signature to $\mathcal{A}$. Again, note that $\mathcal{A}$ can only exploit information leaked from c about the opening value of c_b because the commitment scheme is by assumption hiding.

Note that at this stage, $\mathcal{R}$ cannot request his PCA oracle on (c, r_i), $i \in \{0,1\}$ under the label $m_b \| \Sigma.\mathsf{pk}$. $\mathcal{R}$ would need to query his PCA oracle on such a quantity if he gets a verification (conversion) query on a signature $(c, c_b, -)$ and the message m_b. $\mathcal{R}$ will respond to such a query by simulating the denial protocol (output $\perp$). This simulation differs from the real algorithm when $(c, c_b, -)$ is valid on m_b. Again, such a scenario won't happen with probability at least $(1-\epsilon')^{q_v+q_{sc}}$, because the query would form an existential forgery on the construction since by definition of an invisibility game, $\mathcal{A}$ cannot request $\mathcal{R}$ for a signature on both message m_0, m_1.

The rest of the proof follows directly as in the proof of Theorem 13 . Now, let $\mu = (c, c_b, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c_b))$ be the challenge signature. In case, c is an encryption of $r_{b''}$ (that is if $b' = b''$), then μ corresponds to a valid confirmer signature on m_b. Otherwise, it is not a valid signature on neither m_b nor m_{1-b}. In fact, c_b is a commitment on m_b using a string different from the decryption of c under the label $m_b \| \Sigma.\mathsf{pk}$. Let b_a the bit output by $\mathcal{A}$. $\mathcal{A}$ will output b'' to his challenger in case $b = b_a$ and $1 - b''$ otherwise.

The advantage of $\mathcal{A}$ in such an attack is defined by

$$\epsilon = \mathsf{adv}(\mathcal{A}) = \Pr[b_a = b | b' = b''] - \frac{1}{2}$$

Whereas the advantage of $\mathcal{R}$ is given by

$$\begin{aligned}
\mathsf{adv}(\mathcal{R}) &= (1-\epsilon')^{q_v+q_{sc}} \left[\Pr[b = b_a, b' = b''] + \Pr[b \neq b_a, b' \neq b''] - \frac{1}{2}\right] \\
&= (1-\epsilon')^{q_v+q_{sc}} \left[\Pr[b = b_a | b' = b''] \Pr[b' = b''] + \Pr[b \neq b_a | b' \neq b''] \Pr[b' \neq b''] - \frac{1}{2}\right] \\
&= (1-\epsilon')^{q_v+q_{sc}} \left[\frac{1}{2}(\epsilon + \frac{1}{2}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2}\right] \\
&= \frac{\epsilon}{2}(1-\epsilon')^{q_v+q_{sc}}
\end{aligned}$$

The last but one equation is due to the facts $\Pr[b' \neq b''] = \Pr[b' = b''] = \frac{1}{2}$ as $b'' \overset{R}{\longleftarrow} \{0,1\}$, and to the fact that, in case $b' \neq b''$, the probability that $\mathcal{A}$ answers b is exactly $\frac{1}{2}$ since in that case the challenge signature is not valid on both messages.

□

E Security of the Modified "Signature of a Commitment" Paradigm

E.1 Proof of Theorem 4

The proof is similar to the one of Theorem 12. So we focus on the differences and omit the details.

Proof. (Sketch)

Let $\mathcal{A}$ be an EUF-CMA attacker against the construction. We construct an EUF-CMA attacker $\mathcal{R}$ against the underlying digital signature scheme as follows.

$\mathcal{R}$ gets the parameters of the digital signature from his attacker, and chooses a suitable encryption and commitment scheme. Simulation of the confirmedSign queries (on messages m_i) is done by first computing a commitment c_i on m_i using some random string r_i, then encrypting the string r_i in e_i and finally requesting the challenger for a digital signature σ_i on $e_i \| \diamond \| c_i$. The string (e_i, c_i, σ_i) is output to $\mathcal{A}$ along with a proof of the equality of the decryption of e_i and the opening value of c_i. Such proof can be issued using the cryptosystem private key that $\mathcal{R}$ knows or the randomness used to encrypt r_i in e_i. Confirmation/denial and selective conversion queries can be perfectly simulated with the knowledge of the cryptosystem private key.

At some point, $\mathcal{A}$ will output a forgery $\mu^\star = (e^\star, c^\star, \sigma^\star)$ on some message $m^\star$, which has never been queried before. By definition, $\sigma^\star$ is a valid digital signature on $e^\star \| \diamond \| c^\star$. It will form an existential forgery on the digital signature scheme if $e^\star \| \diamond \| c^\star$ has never been queried before by $\mathcal{R}$ for a digital signature. Suppose there exists $1 \leq i \leq q_s$ such that $e^\star \| \diamond \| c^\star = e_i \| \diamond \| c_i$ where $\mu_i = (e_i, c_i, \sigma_i)$ was the output confirmer signature on the query m_i. Since the ciphertexts (or commitments) do contain the special character $\diamond$, then equality of the strings $e^\star \| \diamond \| c^\star$ and $e_i \| \diamond \| c_i$ implies equality of their prefixes and suffixes, which implies equality of $c^\star$ and c_i. We are then back to Theorem 12. In fact, this equality implies the equality of m_i and $m^\star$ since the used commitment is binding.

□

E.2 Proof of Theorem 5

Proof. Simulation of the key generation is similar to the previous proofs.

For a ConfirmedSign query on a message m_i, the reduction $\mathcal{R}$ (attacker against the cryptosystem) will proceed exactly as a real signer would do, with the exception of maintaining a list of records that contains the queried messages, the output confirmer signatures and the intermediate values used to produce these signatures, namely the random string used in the commitment and the randomness used to encrypt it. This list will be used later for the confirm/deny and selective conversion queries. In fact, for such queries, say (e_i, c_i, σ_i) on m_i, $\mathcal{R}$ will simulate the confirmation protocol (using the rewinding technique or the randomness used to encrypt the opening value of the commitment) if the encryption e_i appears in one record in the list (as an encryption of a string used for commitment), or simulate the denial protocol otherwise. Selective conversion of a confirmer signature whose first field appears in the list is done by revealing the opening value of the commitment, otherwise such a confirmer signature is converted to $\perp$.

The difference of this simulation with the real execution of the algorithm is when a queried signature, say (e_i, c_i, σ_i), is valid but e_i was never used to generate confirmer signatures. We distinguish two cases, either the underlying message m_i has been queried previously on not. In the latter case, such a signature would correspond to an existential forgery on the construction, thus, to an existential forgery on the underlying digital signature. In the former case, the adversary would have to compute a digital signature on $e_i\| \diamond \|c_i$, where e_i was never used before. By the same argument used in the proof of Theorem 4 (for the analysis of the forger's output), we conclude that the adversary would have to compute a digital signature on a string for which he never had obtained a signature. Thus, in the former case, the query would lead to an existential forgery on the underlying signature scheme. Since the latter is by assumption (t, ϵ', q_s)-EUF-CMA secure, the probability that the simulation differs from the real execution is at least $(1-\epsilon')^{q_v+q_{sc}}$.

Finally, in the challenge phase, the adversary outputs two challenging messages m_0, m_1. $\mathcal{R}$ will then produce two strings r_0, r_1 and hands them to his challenger. He gets as a response a challenge ciphertext e on r_b for some $b \in \{0,1\}$. $\mathcal{R}$ will choose a bit $b' \xleftarrow{R} \{0,1\}$ and produces a commitment c on a message $m_{b''}$, for some $b'' \xleftarrow{R} \{0,1\}$, using the string $r_{b'}$. Finally, he will produce a digital signature σ on $e\|\diamond\|c$. The challenge confirmer signature is (e, c, σ). Note, that if $b = b'$, the signature is valid on the message $m_{b''}$, otherwise, it is invalid on both messages. Note also that the adversary exploits only information leaked from the encryption e because the commitment scheme is hiding.

The adversary will continue issuing his queries to $\mathcal{R}$, who will handle them as previously. At the end, the adversary outputs a bit b_a. Clearly the advantage of the adversary is $\epsilon = \Pr[b'' = b_a | b = b'] - \frac{1}{2}$. $\mathcal{R}$ will output b' in case $b'' = b_a$ and $1 - b'$ otherwise.

The advantage of $\mathcal{R}$ is clearly

$$\begin{aligned}
\mathsf{adv}(\mathcal{R}) &= (1-\epsilon')^{q_v+q_{sc}}\left[\Pr[b''=b_a, b'=b] + \Pr[b'' \neq b_a, b' \neq b] - \frac{1}{2}\right]\\
&= (1-\epsilon')^{q_v+q_{sc}}\left[\Pr[b''=b_a | b'=b]\Pr[b'=b] + \Pr[b'' \neq b_a | b' \neq b]\Pr[b' \neq b] - \frac{1}{2}\right]\\
&= (1-\epsilon')^{q_v+q_{sc}}\left[\frac{1}{2}(\epsilon + \frac{1}{2}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2}\right]\\
&= \frac{\epsilon}{2}(1-\epsilon')^{q_v+q_{sc}}
\end{aligned}$$

□

F Efficient Instantiations using Certain Commitments and Cryptosystems

F.1 Proof of Theorem 6

We first remark that the function f used in the definition of the class $\mathbb{C}$ induces a group law in $\mathbb{H} = f(\mathbb{G})$ for the operation $\circ_s$. Moreover, we have $1_{\mathbb{H}} = f(1_{\mathbb{G}})$ and $\forall r \in \mathbb{G}$: $f(r)^{-1} = f(r^{-1})$.

Proof. For completeness, it is clear that if both parties follow the protocol, the prover will always be able to provide a proof that the verifier will accept.

For soundness, we show that the prover can cheat with probability at most 2^{-1} in one round if the verifier chooses b uniformly at random from $\{0,1\}$. In fact, suppose that the prover can answer both challenges for the same commitment t_1. Let r_0 and r_1 be the responses of the prover to the challenges 0 and 1 respectively in Step 3. Since the verifier accepts the proof, we have, $t_1 = f(r_0) \circ_s I = f(r_1)$. Thus, $f(r_1) \circ_s f(r_0)^{-1} = f(r_1 * r_0^{-1}) = I$. Hence, the prover would know a preimage of I. We conclude that a cheating prover can cheat with at most $1/2$, provided f is one-way and the verifier is honest (chooses the bit b uniformly from $\{0,1\}$). Repeating the protocol l times leads to a soundness error which is at most 2^{-l}.

To prove that the proof is ZK, we provide the following simulator.

1. Generate uniformly a random bit $b' \in_R \{0,1\}$. If $b' = 0$, choose $r' \in_R \mathbb{G}$ and sends $t_1 = f(r') \circ_s I$, otherwise, choose $r'' \in_R \mathbb{G}$ and sends $t_1 = f(r'')$ to the verifier.

2. Get b from the verifier. If $b = b'$: if $b = 0$, the simulator sends back r', otherwise, it sends r''. If $b \neq b'$, it goes to Step 1.

The prover's first message is always the function f applied to a random value $r'' \in \mathbb{G}$, and so is the first message of the simulator. Since b' is chosen uniformly at random from $\{0, 1\}$, the probability that the simulator rewinds the verifier is:

$$1 - \Pr[b = b'] = 1 - (\Pr[b = 0, b' = 0] + \Pr[b = 1, b' = 1]) = 1 - (\frac{1}{2}p + \frac{1}{2}(1 - p)) = 1 - \frac{1}{2} = \frac{1}{2}$$

where $p = \Pr[b = 0]$. Therefore, the expected number of rewinds is 2 and as a consequence, the simulator runs in expected linear time. Finally, the distribution of the answers of the prover and of the simulator is again the same. We conclude that the protocol is ZK. It also remains ZK if it is run l times in parallel, where l is either constant or logarithmic in the security parameter. In fact, the simulator of the parallel composition of the protocol will be the parallel composition of the above simulator. Thus, the expected running time of the new simulator is 2^l (probability of not rewinding the verifier is 2^{-l}), which is either constant or polynomial in the security parameter. □

F.2 Proof of Theorem 8

Proof. The confirmation protocol depicted in Figure 3 is a parallel composition of the proofs depicted in Figures 1 and 2. Therefore completeness and soundness follow as a direct consequence of the completeness and soundness of the underlying proofs (see [17]).

To prove that the protocol is ZK. We provide the following simulator (for one execution):

1. Generate $b' \in_R \{0, 1\}$. If $b' = 0$, choose $r' \in_R \mathbb{G}$ and sends $t_1 = f(r') \circ_s I$ and $t_2 = \mathsf{encrypt}(r') \circ_e e$, otherwise, choose $r'' \in_R \mathbb{G}$ and sends $t_1 = f(r'')$ and $t_2 = \mathsf{encrypt}(r'')$ to the verifier.
2. Get b from the verifier. If $b = b'$: if $b = 0$, the simulator sends back r' and the randomness used to encrypt it in $\mathsf{encrypt}(r')$, otherwise, it sends r'' and simulates the proof of t_2 being an encryption of r'' (this proof is simulatable since it is by assumption ZK). If $b \neq b'$, it goes to Step 1.

The prover's first message is an encryption of a random value $r'' \in_R \mathbb{G}$, in addition to $f(r'')$, and so is the simulator's first message. Therefore the distributions of the prover and of the simulator outputs are the same in the first round of the proof. Moreover, the expected number of rewinds is 2 ($\Pr(b \neq b') = \frac{1}{2}$), making the simulator run in expected linear time. The distribution of the prover's messages in the third round is also similar to that of the simulator's messages. We conclude that the confirmation protocol is ZK. Parallel execution of the protocol will remain also ZK if the number of executions l is constant or logarithmic in the security parameter (see the above proof). □

F.3 Proof of Theorem 9

Proof. With the standard techniques, we prove that the denial protocol depicted in Figure 3 is complete and sound with error probability 2^{-l} (l is the number of rounds) provided the verifier is honest and the cryptosystem is one way. Similarly, we provide the following simulator to prove the ZK property.

1. Generate $b' \in_R \{0, 1\}$. If $b' = 0$, choose $r' \in_R \mathbb{G}$ and sends $t_1 = f(r') \circ_s I$ and $t_2 = \Gamma.\mathsf{encrypt}(r') \circ_e e$, otherwise, choose $r'' \in_R \mathbb{G}$ and a random $t_1 \in_R f(\mathbb{G})$ and $t_2 = \Gamma.\mathsf{encrypt}(r'')$.
2. Get b from the verifier. If $b = b'$: if $b = 0$, the simulator sends back r' and the randomness used to encrypt it in $\Gamma.\mathsf{encrypt}(r')$, otherwise, it sends r'' and simulates the proof of t_2 being an encryption of r'' (this proof is simulatable since it is by assumption zero knowledge). If $b \neq b'$, it goes to Step 1.

The prover's first message is an encryption of some random value r'', and the element $t_1 = f(r'' * r^{-1}) \circ_s I$. The simulator's first message is an encryption of a random value r'', and in case $b = 0$ the element $t_1 = f(r'' * r^{-1}) \circ_s I$, whereas in the case $b = 1$, it is the element $t_1 \in_R f(\mathbb{G})$ (independent of r''). Distinguishing these two cases it at least as hard as breaking the IND-CPA security of the underlying cryptosystem. In fact, if the verifier is able to distinguish these two cases, it can be easily used to break the cryptosystem in the IND-CPA sense. Therefore, under the assumption of the IND-CPA security of the cryptosystem, the simulator's and prover's first message distributions are indistinguishable. Moreover, the simulator runs in expected linear time, since the number of rewinds is 2. Moreover, the distribution of the prover's and the simulator's message in the last round are again, by the same argument, indistinguishable under the IND-CPA security of the cryptosystem. Finally, with same argument as above, parallel execution of the protocol remains also ZK if the number of executions is constant or logarithmic in the security parameter. □

G The "Signature of an Encryption" Paradigm

G.1 Proof of Theorem 10

Proof. The adversary $\mathcal{R}$ against the signature underlying the construction will get the parameters of the digital signature he is trying to attack from his challenger. Then, he will choose a suitable cryptosystem. Simulation of signatures is simple; on a query m_i, $\mathcal{R}$ will first compute an encryption c_i of m_i, then request his challenger for a signature on c_i. Let σ_i be the answer of such a query. $\mathcal{R}$ will then output (c_i, σ_i) and produces a ZK proof that c_i decrypts in m_i. Such a proof, in addition to all the proofs involved in the verification/conversion queries is possible for $\mathcal{R}$ to give with the knowledge of the cryptosystem private key.

At some time, the adversary $\mathcal{A}$ against the construction will output a forgery $(c^\star, \sigma^\star)$ on a message $m^\star$, that has never been queried before. $\sigma^\star$ is by definition a digital signature on $c^\star$. The former has never been queried by $\mathcal{R}$ for digital signature, since otherwise $m^\star$ would have been queried before. We conclude that $(c^\star, \sigma^\star)$ is also a valid forgery on the signature scheme. □

G.2 Proof of Theorem 11

Proof. Let $\mathcal{A}$ be the invisibility adversary against the construction, we construct an IND-CPA adversary $\mathcal{R}$ against the underlying cryptosystem as follows.

$\mathcal{R}$ gets the parameters of the target cryptosystem from his challenger, and chooses a suitable digital signature scheme. For a confirmedSign query on m_i, $\mathcal{R}$ will proceed as in the real algorithm, with the exception of maintaining a list $\mathcal{L}$ of records that consists of the query, its encryption, the randomness used to produce the encryption, and finally the digital signature on the encryption. $\mathcal{R}$ can produce digital signatures on any encryption with the knowledge of the signature scheme private key. Moreover, he can confirm any signature he has just generated with the knowledge of the randomness used in the encryption.

For a verification query (c_i, σ_i) on m_i, $\mathcal{R}$ will check $\mathcal{L}$ (after checking of course the validity of σ_i on m_i), if the record $R_i = (m_i, c_i, -, -)$ appears in the list, then he will issue a proof that c_i decrypts in m_i using the third component of the record. Otherwise, he will simulate a proof of the inequality of the decryption of c_i and m_i using the rewinding technique.

For a conversion query, $\mathcal{R}$ will proceed as in a verification query with the exception of providing the non-interactive variant of the proof he would issue if the signature is valid, and the symbol $\perp$ otherwise. This simulation differs from the real one when the queried signature (c_i, σ_i) is valid on m_i however c_i does not appear in the list (as first field of the output confirmer signatures). We distinguish two cases, either the message in question m_i has not been queried before for signature, in which case such a query would correspond to a valid existential forgery on the construction, and thus on the underling signature scheme. Or, the queried signature is on a message that has been queried before, which corresponds to an existential forgery on the underlying signature scheme. Since the signature scheme underlying the construction is (t, ϵ', q_s)-EUF-CMA secure, this scenario does not happen with probability at least $(1 - \epsilon')^{q_v + q_{sc}}$.

At some point, $\mathcal{A}$ produces two messages m_0, m_1. $\mathcal{R}$ will forward the same messages to his challenger and obtain a ciphertext c, encryption of m_b for some $b \overset{R}{\longleftarrow} \{0, 1\}$. $\mathcal{R}$ will produce a digital signature on c and give the result in addition to c to $\mathcal{A}$ as a challenge confirmer signature. It easy to see that $\mathcal{A}$'s answer is sufficient for $\mathcal{R}$ to conclude. Note that after the challenge phase, $\mathcal{A}$ is allowed to issue confirmedSign, verification and conversion queries and $\mathcal{R}$ can handle them as previously.

□