The Joint Signature and Encryption Revisited *

Laila El Aimani †

Abstract

We study the Sign_then_Encrypt, Commit_then_Encrypt_and_Sign, and Encrypt_then_Sign paradigms in the context of three cryptographic primitives, namely designated confirmer signatures, signcryption, and verifiably encrypted signatures. Our study identifies weaknesses in those paradigms which impose the use of expensive encryption (as a building block) in order to meet a reasonable security level. Next, we propose some optimizations which annihilate the found weaknesses and allow consequently cheap encryption without compromising the overall security. Our optimizations further enjoy verifiability, a property profoundly needed in many real-life applications of the studied primitives.

Keywords: Sign_then_Encrypt, Commit_then_Encrypt_and_Sign, Encrypt_then_Sign, (public) verifiability, designated confirmer signatures, signcryption, verifiably encrypted signatures, zero knowledge proofs.

*This paper builds upon the conference papers [29, 31, 30, 32, 33].

†Technicolor, 1 avenue de Belle Fontaine - CS17616, 35576 Cesson-Sévigné, France

Contents

1 Introduction

Cryptographic mechanisms that require both the functionalities of signature and of encryption are becoming nowadays increasingly important. In fact, signatures guarantee the integrity/authenticity of the transmitted data, whereas encryption is needed to ensure either the confidentiality of the signed data or the opacity of the signature. In this document, we study three of such primitives, namely designated confirmer signatures, signcryption, and verifiably encrypted signatures.

Designated confirmer signatures An important feature in digital signatures is the universal verification, i.e. anyone can verify signatures issued by a signer given his public key. However, such a property can be undesirable in some applications and needs to be controlled or limited. A typical example is a software vendor willing to embed signatures in his products such that only paying customers are entitled to check the authenticity of these signatures. Undeniable signatures, introduced in [18], 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 [17], 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. Finally, a desirable property in designated confirmer signatures is the convertibility of the signatures to ordinary ones. Indeed, such a property turned out to play a central role in fair payment protocols [11].

Signcryption This primitive was introduced by Zheng [73] to simultaneously perform the functions of both signature and encryption in a way that is more efficient than signing and encrypting separately. A typical use-case of this mechanism is electronic elections where the voter wants to encrypt his vote to guarantee privacy, and at the same time, the voting center needs to ensure that the encrypted vote comes from the entity that claims to be its provenance. In addition to the signcryption (operation of simultaneously signing and encrypting the data) and unsigncryption (operation of recovering the data and checking its authenticity) procedures, we require in a signcryption scheme the verifiability property; i.e. the possibility to prove efficiently the validity of a given signcryption, or to prove that a signcryption has indeed been produced on a given message. In fact, if we consider the example of electronic elections, the voting center might require from the voter a proof of validity of the "signcrypted" vote[1]. Also, the trusted party (the receiver) that decrypts the vote might be compelled, for instance to resolve some later disputes, to prove that the sender has indeed produced the vote in question; therefore, it would be desirable to support the prover with efficient means to provide such proofs without having to disclose his private input.

Verifiably encrypted signatures (VES) Verifiably encrypted signatures were introduced in [8] for applications such as contract signing; suppose that Alice wants to show Bob that she signed a message, but does not want him to possess her signature on that message. Alice can achieve this by encrypting her signature using the public key of a trusted party, called the *adjudicator*, and sending this to Bob along with a proof that she has given him a valid encryption of her signature. Bob can verify the validity of this verifiably encrypted signature, and may request later the trusted party for the actual signature Alice issued on the message. This property is often referred to in the literature as *extraction*.

1.1 Related work

Since the introduction of the aforementioned primitives, many realizations (of these primitives) which achieve different levels of security have been proposed. On a high level, security in these primitives involves basically two properties; privacy and unforgeability. The last property is analogous to unforgeability in digital signatures and it denotes the difficulty to impersonate the signer. Privacy in case of confirmer signatures (or signcryptions) is similar to indistinguishability in public key encryption, and it refers to the difficulty to distinguish confirmer signatures (signcryptions) based on the underlying messages. In verifiably encrypted signatures, we require only the difficulty to extract the underlying

[1] It is worth noting that this need already manifests in the IACR electronic voting scheme (The Helios voting scheme) where the sender proves the validity of the encrypted vote to the voting manager. Besides, this property can also be needed in filtering out spams in a secure email system.

digital signatures. Defining formally those two properties is a fundamental divergence in constructions realizing these primitives as there are many issues which come into play. One consequential difference between security models is whether the adversary is external or internal to the system. The former case corresponds to *outsider security*, e.g. [27], whereas the latter denotes *insider security* which protects the system protagonists even when some of their fellows are malicious or have compromised/lost their private keys [15, 38, 69, 1, 52]. It is naturally possible to mix these notions into one single scheme, i.e. insider privacy and outsider unforgeability [1, 19], or outsider privacy and insider unforgeability [2]. However, the most frequent mix is the latter as illustrated by the number of works in the literature, e.g. [1, 48, 2]; it is also justified by the necessity to protect the signer from anyone trying to impersonate him including entities in the system. Insider privacy is by contrast needed in very limited applications; the typical example, given in [1], is when the adversary happens to steal the private key of the signer, but we still wish to protect the privacy of the recorded signcryptions/confirmer signatures sent by the genuine signer.

Building complicated systems upon simple and basic primitives is customary in cryptography as it allows to re-use existing work about the primitives, and it achieves easy-to-understand and easy-to-prove systems. The classical constructions used to build the above mentioned primitives are:

1.1.1 Sign_then_Encrypt (StE)

For confirmer signatures, this technique consists in first signing the message in question, then encrypting the produced signature. The construction was first formally [2] described in [15], and it suffered the resort to concurrent zero knowledge (ZK) protocols of general NP statements in the confirmation/denial protocol (i.e. proving knowledge of the decryption of a ciphertext, and that this decryption forms a valid signature on the given message). Later, the proposal in [43] circumvented this 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 [71]; it 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.

For signcryption, this technique consists in similarly signing the message to be signcrypted, however the signcryption corresponds to the encryption of the produced digital signature *in addition to the message*. The construction was first described and analyzed in [1]. It was further extended in [52] to support the multi-user setting, i.e. a setting where many senders interact with many receivers, using tag-based encryption. Finally, there are the recent constructions [19] which achieve multi-user insider security using (tag-based) encryption schemes from the hybrid encryption paradigm. It is worth noting that none of these constructions treat verifiability.

For verifiably encrypted signatures, this technique (EtS) has never been explicitly described as the underlying components (signature and encryption) need to *jointly* satisfy some properties in order to have the encrypted signature publicly verifiable. However, many realizations of VES instantiate this principle; we note as an illustration [8, 63, 74].

1.1.2 Commit_then_Encrypt_and_Sign (CtEaS)

This technique was first described in the context of signcryption in [1]. It has been essentially introduced to parallel encryption and signature. In fact, signcryption of a message using this technique is obtained by committing to the message in question, then encrypting the message and the randomness used to form the commitment, *and* signing the commitment. Later, (a variant of) this technique was adopted in [38] for confirmer signatures, i.e. encryption is performed only on the randomness used to form the commitment, and signature is obtained on this encryption *in addition* to the commitment. This construction was identified to be flawed in [69], where the authors propose to use encryption with labels as building blocks in order to repair the flaw. More precisely, a confirmer signature on a message m is obtained by first committing to m, then encrypting the randomness used in the commitment w.r.t. the label $m \| pk$, pk being the public key of the used signature scheme, *and* finally signing the commitment. Although CtEaS can be used with *any* signature scheme (StE needs to be used with special

[2] The idea without proof was already known, for instance, it was mentioned in [24].

signature schemes in order to allow an efficient verifiability), it is still afflicted with the recourse to general ZK proofs, e.g. proving in concurrent ZK the knowledge of the decryption of an IND-CCA encryption that equals a string used for commitment. An efficient instantiation is however achieved for confirmer signatures in [69] using Camenisch-Shoup's verifiable encryption scheme [16] and Pedersen's commitment scheme.

1.1.3 Encrypt_then_Sign (EtS)

This technique consists in first encrypting the message in question, then producing a signature on this encryption. EtS has been introduced for two-user setting signcryption in [1]. It has been later extended in [52] to support the multi-user setting using tag-based encryption. Besides, in the same work [52], the authors present variations of the paradigm using symmetric primitives and achieve efficient signcryption schemes but the expense of security (outsider unforgeability/privacy) and verifiability.

To summarize the state of the art, StE, CtEaS, and EtS have been studied for the aforementioned primitives in different security models. These studies conclude the need for CCA secure encryption in order to ensure insider privacy. Since *outsider security might be all one needs* for privacy as quoted by the authors in [1], we propose to relax the requirement on insider privacy with the hope of weakening the strong assumption (CCA security) on the encryption. The work [52] achieves some results in this direction as it rests on CPA secure *symmetric* encryption, but at the expense of verifiability.

It would be nice to study these paradigms in the outsider privacy model, and provide efficient variants (if need be) which rest on cheap encryption while providing good verifiability properties. This is the main contribution of this paper.

1.2 Contributions and overview of our techniques

As stated earlier, the main contribution of the present paper is a thorough study of StE, CtEaS, and EtS in the outsider privacy model. Our study concludes that both StE and CtEaS require expensive assumptions on the underlying encryption (PCA security) in order to derive signcryption or confirmer signatures with outsider privacy. We do this by first proving the insufficiency of OW-CCA and NM-CPA secure encryption using the celebrated *meta-reduction* tool, then by exhibiting a simple attack if the system is instantiated from certain encryption schemes. These negative results can be explained by an inherent weakness in these constructions that consists in the possibility of creating confirmer signatures or signcryption without the help of the signer. Next, we propose ameliorations of the paradigms that annihilate this weakness without compromising the security. We achieve this by binding the digital signature to the resulting signcryption/confirmer signature. Consequently, our optimizations of StE and CtEaS (for both signcryption and confirmer signatures) rest on cheap encryption (CPA secure asymmetric encryption) and support efficiently the verifiability property required for such mechanisms. We actually describe explicitly, and for the first time, the verifiability proofs in case the constructions are instantiated from large classes of encryption, commitment, and signature schemes. We have further the following side-results:

- Our negative results for StE serve also for disproving the conjectured security of a well known undeniable signature [24]. Moreover, the adjustment we propose to the basic StE paradigm fixes also this scheme ([24]), and captures further undeniable signatures that were proposed later [50, 65].
- We provide practical instantiations of EtS, in the context of both signcryption and confirmer signatures, which support efficiently verifiability. In fact, some of the required verifiability proofs involve non-interactive proofs of correctness of a decryption. We identify several encryption schemes that efficiently implement this feature.
- We provide a generic construction of VES from StE that captures certain realizations of this primitive. We further prove the insufficiency, using the same techniques developed for signcryption/confirmer signatures, of certain assumptions on the underlying encryption for the opacity of the resulting VES.

2 Preliminaries

2.1 Cryptographic primitives

2.1.1 Digital signatures

A signature scheme 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 [42]. 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.

2.1.2 Public key encryption

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 inverting 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, Plaintext Checking Attack (PCA) in which the adversary is allowed to query an oracle on pairs (m, c) and gets answers whether c encrypts m 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 $\mathsf{goal} \in \{\mathsf{OW}, \mathsf{IND}, \mathsf{NM}\}$ and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{CCA}\}$. We refer to [3] for the formal definitions of these notions as well as for the relations they satisfy.

Tag-based encryption, also referred to as encryption with labels, was first introduced in [67]. In these schemes, the encryption algorithm takes as input, in addition to the public key and the message to be encrypted, a tag which specifies information related to the message and to its encryption context. Similarly, the decryption algorithm takes additionally to the ciphertext and the private key the tag under which the ciphertext was created. Presence of the tag adds further attack models for this type of encryption, e.g. when does the adversary communicate the challenge tag? is he allowed to ask queries w.r.t. this challenge tag? etc. We refer to [67, 49] for the formal security definitions of such a type of encryption.

2.1.3 Key/Data encapsulation mechanisms (KEM/DEMs)

A KEM comprises three algorithms: the key generation algorithm `keygen`, the encapsulation algorithm `encap` and the decapsulation algorithm `decap`. The typical security goals that a KEM should satisfy are similar to those defined for encryption schemes. Similarly, when conjoined with the three attack models CPA, PCA and CCA, they yield nine security notions whose definitions follow word-for-word from the definitions of the encryption schemes notions. A DEM is simply a secret key encryption scheme given by the same algorithms forming a public key encryption scheme. KEMs can be efficiently combined with DEMs to build secure public key encryption schemes. This paradigm is called the hybrid encryption paradigm and we refer to [46] for the necessary and sufficient conditions on the KEMs and the DEMs to obtain a certain level of security for the resulting hybrid encryption scheme.

2.1.4 Commitment schemes

A commitment scheme [12] consists of a key generation algorithm `keygen`, a commitment algorithm `commit`, and an opening algorithm `open`. We require in a commitment scheme the hiding and binding properties. The former informally denotes the difficulty to infer information about the message from the corresponding commitment, whereas the latter denotes the difficulty to come up with collisions, i.e. find two different messages that map to the same value by the `commit` algorithm. A further property might be required, in some applications, for commitments, namely injectivity. It denotes that `commit` for

Prover P		Verifier V
Compute $c = \texttt{Cmt}(x)$	$\xrightarrow{\quad c \quad}$	
	$\xleftarrow{\quad b \quad}$	Choose $b \xleftarrow{R} \{0,1\}^\ell$ $(b \in \mathbb{N})$
Compute $z = \texttt{Rsp}(c, w, b)$	$\xrightarrow{\quad z \quad}$	
		Compute $d = \texttt{Dcd}(x, c, b, z)$

Figure 1: Proof of membership to the language $\{w\colon (x, w) \in \mathcal{R}\}$ Common input: x and Private input : w

a fixed message (viewed as a function of the opening value) is injective: two different opening values lead two different commitments.

2.2 Proofs of Knowledge (PoK)

A proof of knowledge is modeled by an interactive proof system, first introduced in [41]. Such a system informally consists of a prover P trying to convince a verifier V that an instance x belongs to a language L. x refers to the common input whereas $(P, V)(x)$ denotes the proof instance carried between P and V at the end of which V is (not) convinced with the membership of the alleged instance x to L:

$$(P, V)(x) \in \{\mathsf{Accept}, \mathsf{Reject}\}$$

P is modeled by a probabilistic Turing machine whereas V is modeled by a *polynomial* probabilistic Turing machine. During $(P, V)(x)$, the parties exchange a sequence of messages called the proof transcript.
A proof of knowledge should satisfy **completeness** which denotes the property of successfully running the protocol is both parties are honest. A further property required in proofs of knowledge is **soundness** notion which captures the inability of a cheating prover P to convince the verifier V with an invalid statement.

(Concurrent) ZK proofs of knowledge Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **zero knowledge** if for every $x \in L$, the proof transcript $(P, V)(x)$ can be produced by a probabilistic polynomial-time algorithm (in the size of the input) S with indistinguishable probability distributions. A proof of knowledge is said to provide concurrent ZK if the prover interacts concurrently with many verifiers (that potentially might collude) on many instances of the proof. It was shown in [28] that every NP language accepts a concurrent ZK proof system.

Σ **protocols** A *public-coin protocol* is a protocol in which the verifier chooses all its messages randomly from publicly known sets. A *three-move protocol* can be written in a canonical form in which the messages sent in the three moves are often called commitment, challenge, and response respectively. The protocol is said to have the *honest-verifier zero-knowledge property (HVZK)* if there exists an algorithm that is able, provided the verifier behaves as prescribed by the protocol, to produce, without the knowledge of the secret, transcripts that are indistinguishable from those of the real protocol. The protocol is said to have the *special soundness property (SpS) property* if there exists an algorithm that is able to extract the secret from two accepting transcripts of the protocol with the same commitment and different challenges. Finally, a three-move public-coin protocol with both the HVZK and SpS properties is called a Σ *protocol*. In the rest of the document, `Cmt`, `Rsp`, and `Dcd` denote respectively the algorithms used to compute the commitment, the response, and the decision in a Σ protocol.

The transcript tr of the Σ protocol depicted in Figure is formed from the messages exchanged between the prover and the verifier, namely (c, b, z). Finally, we will denote by `trSim` the algorithm that simulates the transcript of a Σ protocol on a given instance.

2.3 Cryptographic reductions

A reduction in cryptology, often denoted $\mathcal{R}$, is informally an algorithm solving a certain problem given access to an adversary $\mathcal{A}$ against a certain cryptosystem. To be able to use $\mathcal{A}$, the reduction must simulate $\mathcal{A}$'s environment (instance generation, queries if any...) in a way that is (almost) indistinguishable from the standard execution of the procedures in question. Both the reduction and the adversary are considered probabilistic Turing machines. The advantage of the reduction $\mathsf{adv}(\mathcal{R})$ is by definition the success probability in solving the given instance of the problem in question where the probability is taken over the instance generation and the random coins of both $\mathcal{R}$ and $\mathcal{A}$. Similarly, the advantage of the adversary $\mathcal{A}$, denoted by $\mathsf{adv}(\mathcal{A})$ refers to the success probability in breaking the cryptosystem in question (the probability being taken over all the coin tosses).

Fact 1 *Let $\mathcal{R}$ be a reduction solving a certain problem given access to an adversary $\mathcal{A}$ against a certain cryptosystem. Let C denote the event corresponding to $\mathcal{R}$ providing a simulation (of $\mathcal{A}$'s environment) that is indistinguishable from the standard one. We have* $\mathsf{adv}(\mathcal{R}) \leq \min(\Pr[\mathsf{C}], \mathsf{adv}(\mathcal{A}))$

Proof Let F be the event corresponding to $\mathcal{A}$ breaking the cryptosystem. We have $\mathsf{adv}(\mathcal{R}) \leq \Pr[\mathsf{F} \wedge \mathsf{C}]$. By definition $\mathsf{adv}(\mathcal{A}) = \Pr[\mathsf{F} \mid \mathsf{C}]$. Thus $\mathsf{adv}(\mathcal{R}) \leq \mathsf{adv}(\mathcal{A}) \Pr[\mathsf{C}]$ which concludes the proof. ∎

2.3.1 Key-preserving reductions

These reductions refer to a wide and popular class of reductions which supply the adversary with the same public key as its challenge key. In this document, we restrict this notion to a smaller class of reductions.

Definition 1 (Key-preserving reductions) *Let $\mathcal{A}$ be an adversary which solves a problem A that is perfectly reducible to OW-CPA breaking a certain public key encryption scheme Γ. Let further $\mathcal{R}$ be a reduction breaking a certain security notion of Γ w.r.t. a public key pk given access to $\mathcal{A}$. $\mathcal{R}$ is said to be* key-preserving *if it launches $\mathcal{A}$ over her own challenge key pk in addition to some other parameters (chosen freely by her) according to the specification of $\mathcal{A}$.*

Such reductions were for instance used in [58] to prove a separation between factoring and IND-CCA-breaking some factoring-based encryption schemes in the standard model.

As mentioned in [58], key-preserving reductions are transitive, i.e. if there is a key-preserving reduction from A to B, and another key-preserving reduction from B to C, then there is a key-preserving reduction from A to C. Finally, it is worth noting that reductions among security notions of a given cryptosystem are key-preserving (in the wide sense of the term).

3 Convertible Designated Confirmer Signatures (CDCS)

3.1 Syntax

A convertible designated confirmer signatures (CDCS) scheme consists of the following procedures:

$\mathtt{setup}(1^\kappa)$ On input a security parameter κ, probabilistically generate the public parameters *param* of the scheme. Although not always explicitly mentioned, *param* will serve as an input to all the algorithms/protocols that follow.

$\mathtt{keygen}_{\mathsf{E}}(1^\kappa, param)$ This probabilistic algorithm outputs the key pair $(pk_{\mathsf{E}}, sk_{\mathsf{E}})$ of the entity E in the system; E can either be the signer S who issues the confirmer signatures, or the confirmer C who confirms/denies the signatures.

$\mathtt{sign}_{\{sks,pkc\}}(m)$ On input sk_{S}, pk_{C} and a message m, outputs a confirmer signature signature μ on m.

$\mathtt{verify}_{\{coins\ \vee\ sk_{\mathsf{C}}\}}(m, \mu, pk_{\mathsf{S}}, pk_{\mathsf{V}})$ This is an algorithm, run by the signer on a *just generated* signature or by the confirmer on *any* signature. The input to the algorithm is, in addition to pk_{S} and pk_{C}, the message and the signature, the coins *coins* used to produce the signature if the algorithm is run by the signer, or sk_{C} if it is run by the confirmer. The output is either 1 if the signature if valid, or 0 otherwise.

$\mathtt{sconfirm}(m, \mu, pk_{\mathsf{S}}, pk_{\mathsf{C}})$ This is an interactive protocol where the signer S convinces a verifier V of the validity of a signature he has just generated. The common input comprises the signature and the message in question, in addition to pk_{S} and pk_{C}. The private input of S consists of the random coins used to produce the signature μ on m.

`confirm/deny`$(m, \mu, pk_{\mathsf{S}}, pk_{\mathsf{C}})$ These are interactive protocols between the confirmer C and a verifier V. Their common input consists of pk_{S}, pk_{C}, the alleged signature μ, and the message m in question. The confirmer uses 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.

`convert`$_{sk_{\mathsf{C}}}(m, \mu, pk_{\mathsf{S}}, pk_{\mathsf{C}})$ This is an algorithm run by the confirmer C using sk_{C}, in addition to pk_{C} and pk_{S}, on a potential confirmer signature μ on m. The result is either $\perp$ if μ is not output by `sign`$_{\{sk_{\mathsf{S}}, pk_{\mathsf{C}}\}}(m)$, or a string σ which is a valid digital signature on m w.r.t. pk_{S}.

`verifyconverted`$_{pk_{\mathsf{S}}}(m, \sigma)$ 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 [71], the author considers a further protocol used by the confirmer to prove the correctness of the conversion. We show in the following that all the constructions provided in this paper extend readily to this augmented model.*

3.2 Security model

A CDCS scheme should meet the following properties:

Completeness. Every signature produced by `sign` should be validated by the algorithm `verify` and correctly converted. Moreover, valid signatures should be correctly confirmed by `sconfirm` and `confirm`, and invalid signatures should be correctly denied by `deny` if the entities in question $\{\mathsf{S}, \mathsf{C}\}$ follow honestly the protocols.

Security for the verifier (soundness). This property informally means that an adversary who compromises the private keys of both the signer and the confirmer cannot convince the verifier of the validity (invalidity) of an invalid (a valid) confirmer signature.

Non-transferability of `sconfirm`*,* `confirm`*, and* `deny`*.* This property informally captures the simulatability of these protocols. More formally, it is defined through the following game which involves the adversary, the signer, the confirmer, and a simulator:

Game 1: the adversary is given the public keys of the signer and of the confirmer, namely pk_{S} and pk_{C} resp. He can then make arbitrary queries of type {`sign`, `sconfirm`} to the signer and of type {`confirm`, `deny`} and `convert` to the confirmer. Eventually, the adversary presents two strings m and μ for which he wishes to carry out, on the common input $(m, \mu, pk_{\mathsf{S}}, pk_{\mathsf{C}})$, the protocol `sconfirm` with the signer (if μ has been just generated by the signer on m), or the protocols {`confirm`, `deny`} with the confirmer. The private input of the signer is the randomness used to generate the signature μ (in case μ is a signature just generated by the signer), whereas the private input of the confirmer is his private key sk_{C}. The adversary continues issuing queries to both the signer and the confirmer until he decides that this phase is over and produces an output.

Game 2: this game is similar to the previous one with the difference of playing a simulator instead of running the real signer or the real confirmer when it comes to the interaction of the adversary with the signer in `sconfirm` or with the confirmer in {`confirm`, `deny`} on the common input $(m, \mu, pk_{\mathsf{S}}, pk_{\mathsf{C}})$. The simulator is not given the private input of neither the signer nor the confirmer. It is however allowed to issue a single oracle call that tells whether μ is a valid confirmer signature on m w.r.t. pk_{S} and pk_{C}. Note that the simulator in this game refers to a probabilistic polynomial Turing machine with rewind.

The signatures issued by are said to be non-transferable if there exists an efficient simulator such that for all $(pk_{\mathsf{S}}, pk_{\mathsf{C}})$, the outputs of the adversary in Game 1 and Game 2 are indistinguishable. In other words, the adversary should not be able to tell whether he is playing **Game 1** or **Game 2**. Note that this definition achieves only the so-called *offline non-transferability*, i.e. the adversary is not supposed to interact concurrently with the prover and an unexpected verifier. We refer to Remark 2 for the details.

Security for the signer (unforgeability). It is defined through the following game: the adversary $\mathcal{A}$ gets the signer's public key pk_{S}, and generates the confirmer's key pair (sk_{C},pk_{S}). $\mathcal{A}$ is further allowed to

Experiment $\mathbf{Exp}^{\mathsf{completeness}}(1^\kappa)$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \mathtt{keygen}_\mathsf{S}(1^\kappa)$; $(pk_\mathsf{C}, sk_\mathsf{C}) \leftarrow \mathtt{keygen}_\mathsf{C}(1^\kappa)$;
3. $m \xleftarrow{R} \mathsf{M}$; $(\mu, coins_\mu) \leftarrow \mathtt{sign}_{\{sk_\mathsf{S}, pk_\mathsf{C}\}}(m)$; $\psi \xleftarrow{R} \mathsf{CS}$;
4. $out_0 \leftarrow \mathtt{verify}_{\{pk_\mathsf{S}, sk_\mathsf{C}, pk_\mathsf{C}\}}(\mu, m)$;
5. $\langle done \mid out_1 \rangle \leftarrow \mathtt{sconfirm}_{\langle \mathsf{V}, \mathsf{S}(coins_\mu)\rangle}(m, \mu, pk_\mathsf{S}, pk_\mathsf{C})$;
6. $\langle done \mid out_2 \rangle \leftarrow \mathtt{confirm}_{\rangle \leftarrow \langle \mathsf{V}, \mathsf{C}(sk_\mathsf{C})\rangle}(m, \mu, pk_\mathsf{S}, pk_\mathsf{C})$;
7. $\langle done \mid out_3 \rangle \leftarrow \mathtt{deny}_{\langle \mathsf{V}, \mathsf{C}(sk_\mathsf{C})\rangle}(m, \psi, pk_\mathsf{S}, pk_\mathsf{C})$;
8. $\sigma \leftarrow \mathtt{convert}_{sk_\mathsf{C}}(\mu, m)$; $out_4 \leftarrow \mathtt{verifyconverted}_{pk_\mathsf{C}}(\sigma, m)$;
9. If $(out_0 = out_1 = out_2 = out_3 = out_4 = 1)$ return 1 else return 0.

Experiment $\mathbf{Exp}^{\mathsf{soundness}}(1^\kappa)$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \mathtt{keygen}_\mathsf{S}(1^\kappa)$; $(pk_\mathsf{C}, sk_\mathsf{C}) \leftarrow \mathtt{keygen}_\mathsf{C}(1^\kappa)$;
3. $(m, \psi) \leftarrow \mathcal{A}(sk_\mathsf{S}, sk_\mathsf{C})$; $(\mu, coins_\mu) \leftarrow sign_{\mathcal{A}(sk_\mathsf{S})}(m)$;
4. $\langle done \mid out_1 \rangle \leftarrow \mathtt{sconfirm}_{\langle \mathsf{V}, \mathcal{A}(sk_\mathsf{S}, sk_\mathsf{C}, coins_\mu)\rangle}(m, \psi, pk_\mathsf{S}, pk_\mathsf{C})$;
5. $\langle done \mid out_2 \rangle \leftarrow \mathtt{confirm}_{\langle \mathsf{V}, \mathcal{A}(sk_\mathsf{S}, sk_\mathsf{C}, coins_\mu)\rangle}(m, \psi, pk_\mathsf{S}, pk_\mathsf{C})$;
6. $\langle done \mid out_3 \rangle \leftarrow \mathtt{deny}_{\langle \mathsf{V}, \mathcal{A}(sk_\mathsf{S}, sk_\mathsf{C}, coins_\mu)\rangle}(m, \mu, pk_\mathsf{S}, pk_\mathsf{C})$;
7. If $(out_1 = out_2 = out_3 = 0)$ return 1 else return 0.

Experiment $\mathbf{Exp}^{\mathsf{non-transferability}}(1^\kappa)$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \mathtt{keygen}_\mathsf{S}(1^\kappa)$; $(pk_\mathsf{C}, sk_\mathsf{C}) \leftarrow \mathtt{keygen}_\mathsf{C}(1^\kappa)$;
3. $(m, \mu) \leftarrow \mathcal{A}^{\mathfrak{S}, \mathfrak{V}, \mathfrak{V}}(pk_\mathsf{S}, pk_\mathsf{C})$
   $\mathfrak{S} : m_i \longmapsto \mathtt{sign}\{sk_\mathsf{S}, pk_\mathsf{C}\}(m_i)$
   $\mathfrak{Cv} : (m_i, \mu_i) \longmapsto \mathtt{convert}_{sk_\mathsf{C}}(m_i, \mu_i)$
   $\mathfrak{V} : (m_i, \mu_i) \longmapsto \{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}(m_i, \mu_i, pk_\mathsf{S}, pk_\mathsf{C})$
4. $b \xleftarrow{R} \{0, 1\}$;
   $\mathtt{if}\ b = 1\ \mathtt{then}\ \langle done \mid out_0 \rangle \leftarrow \mathtt{prove}_{\langle \mathcal{A}, \mathsf{P}(sk_\mathsf{P})\rangle}(m, \mu, pk_\mathsf{S}, pk_\mathsf{C})$;
   $\mathtt{if}\ b = 0\ \mathtt{then}\ \langle done \mid out_1 \rangle \leftarrow \mathtt{prove}_{\langle \mathcal{A}, \mathsf{Sim}\rangle}(m, \mu, pk_\mathsf{S}, pk_\mathsf{C})$;
   if $\mathtt{prove} = \mathtt{sconfirm}\ \mathtt{then}\ \mathsf{P} =$ and $sk_\mathsf{P} = coins_\mu$
   if $\mathtt{prove} \in \{\mathtt{confirm}, \mathtt{deny}\}\ \mathtt{then}\ \mathsf{P} = \mathsf{C}$ and $sk_\mathsf{P} = sk_\mathsf{C}$
5. $b^\star \leftarrow \mathcal{A}^{\mathfrak{S}, \mathfrak{V}, \mathfrak{V}}(m, \mu, pk_\mathsf{S}, pk_\mathsf{C})$
6. If $(b = b^\star)$ and $(out_0 = out_1)$ return 1 else return 0.

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, wins the above game with probability greater than ϵ, where the probability is taken over all the random choices.

Security for the confirmer (invisibility). Invisibility against a chosen message attack (INV-CMA) is defined through the following game between an attacker $\mathcal{A}$ and her challenger $\mathcal{R}$: after $\mathcal{A}$ gets the public parameters of the scheme from $\mathcal{R}$, she starts **Phase 1** where she queries the $\mathtt{sconfirm}$, $\{\mathtt{confirm}, \mathtt{deny}\}$, and $\mathtt{convert}$ oracles in an adaptive way. Once $\mathcal{A}$ decides that **Phase 1** is over, she outputs two messages m_0, m_1 as challenge messages. $\mathcal{R}$ 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 $(m_i, \mu^\star)$, $i = 0, 1$, to the $\mathtt{sconfirm}$, $\{\mathtt{confirm}, \mathtt{deny}\}$, and $\mathtt{convert}$ oracles. At the end, $\mathcal{A}$ outputs a bit b'. She wins the game if $b = b'$. We define $\mathcal{A}$'s advantage as $(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$, where the probability is taken over all the random coins. Finally, a CDCS scheme is $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure if no adversary operating in time t, issuing q_s queries to the signing oracle (followed potentially by queries to the $\mathtt{sconfirm}$ 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.

We have the following remarks regarding our security model:

Remark 2 • *Our definitions of completeness and security for the verifier are the same as those provided in [15, 38, 69].*

Experiment $\mathbf{Exp}^{\text{unforgeability}}(1^\kappa)$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \mathtt{keygen}_\mathsf{S}(1^\kappa)$;
3. $(pk_\mathsf{C}, sk_\mathsf{C}) \leftarrow \mathcal{A}$;
4. $(m^\star, \mu^\star) \leftarrow \mathcal{A}^{\mathfrak{S}}(pk_\mathsf{S}, pk_\mathsf{C}, sk_\mathsf{C})$

   $\mathfrak{S} : m \longmapsto \mathtt{sign}_{\{sk_\mathsf{S}, pk_\mathsf{C}\}}(m)$
5. return 1 if and only if:
   - $\mathtt{verify}_{\{pk_\mathsf{S}, pk_\mathsf{C}, sk_\mathsf{C}\}}[m^\star, \mu^\star] = 1$
   - $m^\star$ was not queried to $\mathfrak{S}$

Experiment $\mathbf{Exp}^{\text{invisibility}}(1^\kappa)$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \mathtt{keygen}_\mathsf{S}(1^\kappa)$; $(pk_\mathsf{C}, sk_\mathsf{C}) \leftarrow \mathtt{keygen}_\mathsf{C}(1^\kappa)$;
3. $(m_0^\star, m_1^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\text{find}, pk_\mathsf{S}, pk_\mathsf{C})$

   $\mathfrak{S} : m \longmapsto \mathtt{sign}_{\{sk_\mathsf{S}, pk_\mathsf{C}\}}(m)$

   $\mathfrak{Cv} : (m, \mu) \longmapsto \mathtt{convert}_{sk_\mathsf{C}}(m, \mu)$

   $\mathfrak{V} : (m, \mu) \longmapsto \{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}(m, \mu, pk_\mathsf{C}, pk_\mathsf{S})$
4. $b \xleftarrow{R} \{0, 1\}$; $\mu^\star \leftarrow \mathtt{sign}\{sk_\mathsf{S}, pk_\mathsf{S}, pk_\mathsf{C}\}(m_b^\star)$
5. $b^\star \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\text{guess}, \mathcal{I}, \mu^\star, pk_\mathsf{S}, pk_\mathsf{C})$

   $\mathfrak{S} : m \longmapsto \mathtt{sign}\{sk_\mathsf{S}, pk_\mathsf{C}\}(m)$

   $\mathfrak{Cv} : (m, \mu)(\neq (m_i^\star, \mu^\star), i = 0, 1) \longmapsto \mathtt{convert}_{sk_\mathsf{C}}(m, \mu)$

   $\mathfrak{V} : (m, \mu)(\neq (m_i^\star, \mu^\star), i = 0, 1) \longmapsto \{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}(m, \mu, pk_\mathsf{C}, pk_\mathsf{S})$
6. If $(b = b^\star)$ return 1 else return 0.

- *Our definition of non-transferability is the same adopted in [15, 38, 69]. In particular, it thrives on the* concurrent *zero knowledgeness of the* {sconfirm, confirm, deny} *protocols, and guarantees only the so-called* offline non-transferability. *In fact, non-transferability is not preserved, as remarked by [51], if the verifier interacts concurrently with the prover and an unexpected verifier. One way to circumvent this shortcoming consists in requiring the mentioned protocols to be* designated verifier proofs of knowledge [47], *i.e. require the verifier to be able to efficiently provide the proofs in question (that underlay* sconfirm, confirm, *and* deny *resp.) such that no efficient adversary is able to tell whether he is interacting with the genuine prover or with the verifier. This approach was adhered to for instance in [20, 54]. We will show that our proposed practical realizations of confirmer signatures satisfy also this stronger notion of* online non-transferability, *as the underlying* {sconfirm, confirm, deny} *are* Σ *protocols that can be efficiently turned into designated verifier proofs.*
- *We consider the* insider security model against malicious confirmers *in our definition for unforgeability. I.e. the adversary is* allowed *to choose his key pair* $(sk_\mathsf{C}, pk_\mathsf{C})$. *This is justified by the need of preventing the confirmer from impersonating the signer by issuing valid signatures on his behalf. Hence, our definition of unforgeability, which is the same as the one considered by [71], implies its similars in [15, 38, 69].*
- *Our definition of invisibility, namely INV-CMA, is considered in the* outsider security model. *I.e., the adversary does not know the private key of the signer. Actually, outsider security can be enough in many situations as argued earlier in the introduction. Moreover, our definition of invisibility, oppositely to the definitions in [38, 69], allows the signer to sign the same message many times without loss of invisibility, which is profoundly needed in licensing software. Finally, we note that the notion INV-CMA does not guarantee the non-transferability of the signatures. I.e., the confirmer signature might convince the recipient that the signer was involved in the signature of some message. We refer to the discussion in [38] (Section 3) for techniques that can be used by the signer to camouflage the presence of valid signatures.*

3.3 Classical constructions for confirmer signatures

Let Σ be a digital signature scheme given by Σ.keygen which generates a key pair ($\Sigma.sk$, $\Sigma.pk$), Σ.sign, and Σ.verify. Let furthermore Γ denote a public key encryption scheme described by Γ.keygen that generates the key pair ($\Gamma.sk$,$\Gamma.pk$), Γ.encrypt, and Γ.decrypt. Finally, let Ω be a commitment scheme given by the algorithms Ω.commit and Ω.open. The most popular paradigms used to devise confirmer signature schemes from basic primitives are (in all these paradigms, ($\Sigma.pk$,$\Sigma.sk$) forms the signer's key pair, whereas ($\Gamma.pk$,$\Gamma.sk$) forms the confirmer's key pair).

3.3.1 The *"sign-then-encrypt" (StE) paradigm*

Given a message m, sign first produces a signature σ on the message using $\Sigma.sk$, then encrypts it under $\Gamma.pk$. The result forms the confirmer signature on m. To convert such a signature, one simply decrypts it using $\Gamma.sk$ in σ (σ must be a valid digital signature on m w.r.t. $\Sigma.pk$). The sconfirm, confirm, and deny protocols, on the common input $(\mu, m, \Gamma.pk, \Sigma.pk, pk_\mathsf{V})$, boil down to the following *concurrent* zero-knowledge proofs:

$$\texttt{sconfirm} = \mathsf{PoK}\{(\sigma, coins_\mu) : \mu = \Gamma.\texttt{encrypt}_{\Gamma.pk}(\sigma, coins_\mu) \wedge \Sigma.\texttt{verify}_{\Sigma.pk}(\sigma, m) = 1\}$$

$$\texttt{confirm} = \mathsf{PoK}\{(\sigma, \Gamma.sk) : \sigma = \Gamma.\texttt{decrypt}_{\Gamma.sk}(\mu) \wedge \Sigma.\texttt{verify}_{\Sigma.pk}(\sigma, m) = 1\}$$

$$\texttt{deny} = \mathsf{PoK}\{(\sigma, \Gamma.sk) : \sigma = \Gamma.\texttt{decrypt}_{\Gamma.sk}(\mu) \wedge \Sigma.\texttt{verify}_{\Sigma.pk}(\sigma, m) \neq 1\}$$

The above languages are in NP and thus accept efficient concurrent zero-knowledge proof systems [28] which guarantees the completeness, soundness, and (offline) non-transferability of the resulting signatures.

3.3.2 The *"encrypt-then-sign" (EtS) paradigm*

Given a message m, `sign` produces an encryption e on m using $\Gamma.pk$, then produces a signature σ on e using $\Sigma.sk$; the confirmer signature is the pair (e, σ). Similarly, `sconfirm`, `confirm`, and `deny` amount to *concurrent zero-knowledge proofs knowledge*. Finally, `convert` outputs a zero knowledge non-interactive (NIZK) proof that m is the decryption of e; such a proof is feasible since the statement in question is in NP ([40] and [5]).

3.3.3 The *"commit-then-encrypt-and-sign" (CtEaS) paradigm*

This construction has the advantage of performing the signature and the encryption *in parallel* in contrast to the previous sequential compositions. It includes among its building blocks, contrarily to the previous constructions, a public key encryption scheme Γ' that supports labels. To sign a message m, one first produces a commitment c on it using some random nonce r, then encrypts r (using $\Gamma'.pk$), under the label $m\|\Sigma.pk$, *and* produces a signature σ on c using $\Sigma.sk$. The signcryption is the triple (e, c, σ). Confirmation (with either the confirmer or the signer) and denial are similarly achieved by proving the underlying statements in concurrent zero-knowledge. Finally, conversion of a valid signature (e, c, σ) on some message consists in outputting the decryption of e.

Remark 3 *It is possible to require a proof in the* `convert` *algorithms of StE and CtEaS, that the revealed information is indeed a correct decryption of the encryption in question; such a proof is again possible to issue (with or without interaction) since the corresponding statement is in NP.*

4 Negative Results for CDCS

In this section, we show that StE and CtEaS require at least IND-PCA encryption in order to lead to INV-CMA secure confirmer signatures. We proceed as follows.

First, we rule out the OW-CPA, OW-PCA, and IND-CPA notions by remarking that ElGamal's encryption meets all those notions (under different assumptions), but cannot be employed in either StE or CtEaS. In fact, the invisibility adversary (against the confirmer signature) can create a new confirmer signature (by re-encrypting the ElGamal encryption) on the challenge message, and query it for conversion or verification. The answer of such a query is sufficient to conclude. Actually, the attack we exhibit applies to a wide class of encryption schemes that we call *homomorphic encryption*.

Next, we show the insufficiency of OW-CCA and NM-CPA encryption by means of efficient meta-reductions which forbid the existence of reductions from the invisibility of the resulting confirmer signatures to the OW-CCA or NM-CPA security of the underlying encryption. We first show this impossibility result for a specific kind of reductions, then we extend it to arbitrary reductions assuming further complexity assumptions on the used encryption scheme.

4.1 A breach in invisibility using homomorphic encryption

Definition 2 (Homomorphic encryption) *A partially homomorphic public key encryption scheme Γ given by* $\Gamma.$`keygen`, $\Gamma.$`encrypt`, *and* $\Gamma.$`decrypt` *has the following properties:*

1. *The message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$ are groups w.r.t. some binary operations $*_e$ and $\circ_e$ respectively.*

2. $\forall(sk, pk) \leftarrow \Gamma.\texttt{keygen}(1^\kappa)$ *for some security parameter* κ, $\forall m, m' \in \mathcal{M}$:

$$\Gamma.\texttt{encrypt}_{pk}(m *_e m') = \Gamma.\texttt{encrypt}_{pk}(m) \circ_e \Gamma.\texttt{encrypt}_{pk}(m').$$

Examples of partially homomorphic encryptions in the literature include ElGamal [34], Boneh-Boyen-Shacham [7], and Paillier [55]. All those schemes are IND-CPA secure. In the rest of this document, we refer to such schemes as homomorphic[3] encryption schemes.

Remark 4 *Homomorphic encryption with labels is similarly defined; i.e. the message space* $\mathcal{M}$ *and the ciphertext space* C *are groups w.r.t. some binary operations* $*_e$ *and* $\circ_e$ *respectively. Moreover,* $\forall(sk, pk) \leftarrow \Gamma.\texttt{keygen}(1^\kappa)$ *for some security parameter* κ*, for all labels* L, $\forall m, m' \in \mathcal{M}$:

$$\Gamma.\texttt{encrypt}_{pk}(m *_e m', L) = \Gamma.\texttt{encrypt}_{pk}(m, L) \circ_e \Gamma.\texttt{encrypt}_{pk}(m', L).$$

Fact 2 *The StE (CtEaS) paradigm cannot lead to INV-CMA secure confirmer signatures when used with homomorphic encryption (with labels).*

Proof Let m_0, m_1 be the challenge messages the invisibility adversary $\mathcal{A}$ outputs to his challenger. We show that the mentioned paradigms are prone to these simple attacks if the underlying encryption scheme is homomorphic:

- *StE paradigm.* Let Γ and Σ denote respectively the encryption and signature schemes used as constituents. $\mathcal{A}$ will receive as a challenge confirmer signature a certain $\mu_b = \Gamma.\texttt{encrypt}(\Sigma.\texttt{sign}(m_b))$, where $b \xleftarrow{R} \{0, 1\}$ and his task is to guess correctly the used b. To solve his challenge, $\mathcal{A}$ will obtain another encryption, say $\widetilde{\mu_b}$, of $\texttt{sign}(m_b)$ by multiplying μ_b with an encryption of the identity element (of the message space of Γ). According to the invisibility experiment, $\mathcal{A}$ can query $\widetilde{\mu_b}$ for conversion or verification (w.r.t. either m_0 or m_1) and the answer to such a query is sufficient for $\mathcal{A}$ to conclude.
- *CtEaS paradigm.* Let Γ, Σ, and Ω denote respectively the encryption (with labels), signature, and commitment schemes used as building blocks. $\mathcal{A}$ gets as a challenge confirmer signature a certain $\mu_b = [e, c = \Omega.\texttt{commit}(m_b, r), \Sigma.\texttt{sign}(c)]$ ($b \in \{0, 1\}$) where e is an encryption of r w.r.t. the label $m \| \Sigma.pk$. Similarly, $\mathcal{A}$ will compute a new confirmer signature of m_b by multiplying e with an encryption of the identity element (of the message space of Γ) w.r.t. the same label $m \| \Sigma.pk$. $\mathcal{A}$ will then query this new signature (w.r.t. either m_0 or m_1) for conversion or verification, and the answer of the latter is sufficient for $\mathcal{A}$ to solve his challenge.

Remark 5 *Note that the EtS paradigm is resilient to the previous attack since the adversary would need to compute a valid digital signature on the newly computed encryption. This is not plausible in the invisibility game (we consider* outsider *invisibility).*

Invisibility in confirmer signatures from EtS (CtEaS) cannot rest on OW-CPA, OW-PCA, or IND-CPA encryption (with labels).

Proof *StE paradigm.* ElGamal's encryption [34] is homomorphic and meets the OW-CPA, OW-PCA, and IND-CPA security notions (under different assumptions).

- *CtEaS paradigm.* Again ElGamal's encryption [34] is a particular homomorphic encryption with labels, where the encryption of a certain message w.r.t. a certain label corresponds to its standard ElGamal encryption (by ignoring the label). It is easy to see that this encryption conserves the OW-CPA, OW-PCA, and IND-CPA security (under the same assumptions) met by the original scheme.

The rest follows from the previous fact. ∎

4.2 Impossibility results for key-preserving reductions

In this paragraph, we prove that NM-CPA and OW-CCA encryption are insufficient for invisible confirmer signatures from StE or CtEaS, if we consider a certain type of reductions. We do this by means of efficient *meta-reductions* that use such reductions (the algorithm reducing NM-CPA (OW-CCA) breaking the underlying encryption scheme to breaking the invisibility of the construction) to break the NM-CPA

[3] These schemes are not to confuse with the so-called *fully homomorphic* encryption which preserves the entire ring structure of the plaintexts (supports both addition and multiplication).

Experiment $\mathbf{Exp}^{\mathsf{WINV\text{-}CMA}}(1^\kappa)$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \mathtt{keygen}_\mathsf{S}(1^\kappa)$; $(pk_\mathsf{C}, sk_\mathsf{C}) \leftarrow \mathtt{keygen}_\mathsf{C}(1^\kappa)$;
3. $(m_0^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\mathsf{find}, pk_\mathsf{S}, pk_\mathsf{C}, sk_\mathsf{V})$
   $\mathfrak{S} : m \longmapsto \mathtt{sign}_{\{sk_\mathsf{S}, pk_\mathsf{C}\}}(m)$
   $\mathfrak{Cv} : (m, \mu) \longmapsto \mathtt{convert}_{sk_\mathsf{C}}(m, \mu)$
   $\mathfrak{V} : (m, \mu) \longmapsto \{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}(m, \mu, pk_\mathsf{C}, pk_\mathsf{S})$
4. $m_1^\star \xleftarrow{R} \mathsf{M}$; $b \xleftarrow{R} \{0,1\}$; $\mu^\star \leftarrow \mathtt{sign}\{sk_\mathsf{S}, pk_\mathsf{S}, pk_\mathsf{C}\}(m_b^\star)$
5. $b^\star \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\mathsf{guess}, \mathcal{I}, \mu^\star, pk_\mathsf{S}, pk_\mathsf{C})$
   $\mathfrak{S} : m \longmapsto \mathtt{sign}\{sk_\mathsf{S}, pk_\mathsf{C}\}(m)$
   $\mathfrak{Cv} : (m, \mu)(\neq (m_0^\star, \mu^\star)) \longmapsto \mathtt{convert}_{sk_\mathsf{C}}(m, \mu)$
   $\mathfrak{V} : (m, \mu)(\neq (m_0^\star, \mu^\star)) \longmapsto \{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}(m, \mu, pk_\mathsf{C}, pk_\mathsf{S})$
6. If $(b = b^\star)$ return 1 else return 0.

(OW-CCA) security of the encryption scheme. Thus, under the assumption that the encryption scheme is NM-CPA (OW-CCA) secure, the meta reductions forbid the existence of such reductions. In case the encryption scheme is not NM-CPA (OW-CCA) secure, such reductions will be useless.

Note that meta-reductions have been successfully used in a number of important cryptographic results, e.g. the result in [10] which proves the impossibility of reducing factoring to the RSA problem, or the results in [57, 56] which show that some well known signatures which are proven secure in the random oracle cannot conserve the same security in the standard model. Such impossibility results are in general partial as they apply only for certain reductions. Our result is in a first stage also partial since it requires the reduction $\mathcal{R}$, trying to attack a certain property of an encryption scheme given by the public key $\Gamma.pk$, to provide the adversary against the confirmer signature with the confirmer public key $\Gamma.pk$. In other terms, our result applies for key-preserving reductions (see Definition 1). Our restriction to such a class of reductions is not unnatural since, to our best knowledge, all the reductions basing the security of the generic constructions of confirmer signatures on the security of their underlying components, feed the adversary with the public keys of these components (signature schemes, encryption schemes, and commitment schemes). Next, we use similar techniques to [58] to extend our impossibility results to arbitrary reductions.

The rest of this subsection will be organized as follows. We first introduce a weak notion of invisibility for confirmer signatures, denoted WINV-CMA, which informally captures the difficulty to distinguish confirmer signatures on an adversarially chosen message from confirmer signatures on a random message (hidden from the adversary). Then we provide our first impossibility results which forbid the existence of key-preserving reductions from NM-CPA (OW-CCA) breaking the encryption underlying the StE or CtEaS constructions to WINV-CMA breaking the constructions themselves. In the next subsection, we extend this impossibility result to arbitrary reductions at the expense of further assumptions on the used encryption.

4.2.1 Weak invisibility (WINV-CMA)

The WINV-CMA notion is defined along the same lines of INV-CMA except at the challenge phase; in this notion the adversary outputs a message $m^\star$ and gets as challenge a confirmer signature $\mu^\star$ on either $m^\star$ or a random message generated by the challenger. There is the natural restriction of not querying the pair $(m^\star, \mu^\star)$ to the `sconfirm`, {`confirm`, `deny`}, and `convert` oracles.

It is obvious that an adversary against WINV-CMA can be easily transformed into an adversary against the INV-CMA property of the same construction. Therefore proving the inexistence of a (key-preserving) reduction from OW-CCA or NM-CPA breaking the encryption underlying the construction to WINV-CMA breaking the construction itself induces the inexistence of a (key-preserving) reduction from OW-CCA or NM-CPA breaking the encryption underlying the construction to INV-CMA breaking the construction.

Lemma 1 *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts a WINV-CMA adversary $\mathcal{A}$ against confirmer signatures from the StE (CtEaS) paradigm to a OW-CCA adversary against the underlying encryption* *scheme. Then, there exists a meta-reduction $\mathcal{M}$ that OW-CCA breaks the encryption scheme in question.*

Let us first interpret this result. This lemma claims that under the assumption of the underlying encryption scheme being OW-CCA secure, there exists no key-preserving reduction $\mathcal{R}$ that reduces OW-CCA breaking the encryption scheme in question to WINV-CMA breaking the construction (from either StE or CtEaS), or if there exists such an algorithm, then the underlying encryption scheme is not OW-CCA secure, thus rendering such a reduction useless.

Proof Let $\mathcal{R}$ be the key-preserving reduction that reduces OW-CCA breaking the encryption scheme underlying the construction to WINV-CMA breaking the construction itself. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to OW-CCA break the same encryption scheme by simulating an execution of the WINV-CMA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack w.r.t. the key pk. $\mathcal{M}$ proceeds as follows:

- *StE paradigm.* Let c be the OW-CCA challenge $\mathcal{M}$ is asked to resolve. $\mathcal{M}$ launches $\mathcal{R}$ over Γ under the same key pk and the same challenge c. Obviously, all decryption queries made by $\mathcal{R}$ can be perfectly answered using $\mathcal{M}$'s challenger (since they are different from the challenge c). $\mathcal{M}$ needs now to simulate a WINV-CMA adversary $\mathcal{A}$ to $\mathcal{R}$. To do so, $\mathcal{M}$ outputs a message $m^\star$ and receives a challenge $\mu^\star$ which is either a valid confirmer signature on $m^\star$ or a confirmer signature on a random message chosen by $\mathcal{R}$. Two cases manifest, either $\mu^\star$ is different from c, in which case $\mathcal{M}$ can perfectly solve his challenge by querying his own challenger for the decryption of $\mu^\star$ and then checking whether the result is a valid digital signature on $m^\star$ or not. Or $\mu^\star$ is equal to c; in this case $\mathcal{M}$ will answer that $\mu^\star$ is not a valid confirmer signature on $m^\star$. In fact, with overwhelming probability, the OW-CCA challenge c is not a valid confirmer signature on a message chosen later by $\mathcal{M}$ (upon receipt of the challenge). Therefore, $\mathcal{M}$ (in both cases) can correctly simulate the WINV-CMA attacker to $\mathcal{R}$.
- *CtEaS paradigm.* $\mathcal{M}$ launches $\mathcal{R}$ over Γ with the same public key pk. After $\mathcal{M}$ gets the label L on which $\mathcal{R}$ wishes to be challenged, he ($\mathcal{M}$) forwards it to his own challenger. Finally, $\mathcal{M}$ gets a challenge ciphertext e, that he forwards to $\mathcal{R}$. Note that $\mathcal{M}$ is allowed to query his own challenger (for decryption) on any pair (ciphertext,label) except on the pair (e, L). Thus, all decryption queries made by $\mathcal{R}$, which are by definition different from the challenge (e, L), can be forwarded to $\mathcal{M}$'s own challenger. At some point, $\mathcal{M}$, acting as a WINV-CMA attacker against the construction, will output a messages $m^\star$ such that $L \neq m^\star \| \Sigma.pk$, where $\Sigma.pk$ is the public key of the digital signature underlying the construction. $\mathcal{M}$ gets as response a challenge $\mu^\star = (e^\star, c^\star, \sigma^\star)$ which is a confirmer signature on either $m^\star$ or a message chosen by $\mathcal{R}$. If $\mu^\star$ is a valid confirmer signature on $m^\star$, then $e^\star$ encrypts some $r^\star$ under pk w.r.t. the label $m^\star \| \Sigma.pk$ such that $c^\star$ is a valid commitment on $m^\star$ using $r^\star$, and $\sigma^\star$ is a valid digital signature on $c^\star$ w.r.t. $\Sigma.pk$. Thus, to solve his challenge, $\mathcal{M}$ requests his own challenger for the decryption of $e^\star$ w.r.t. the label $m^\star \| \Sigma.pk$ ($\mathcal{M}$ is allowed to issue such a query even if $e = e^\star$ since he chose $m^\star$ such that $L \neq m^\star \| \Sigma.pk$). The answer of such a query will allow $\mathcal{M}$ (behaving as a WINV-CCA attacker) to perfectly answer his invisibility challenge.

To sum up, $\mathcal{M}$ is able to perfectly answer the decryption queries made by $\mathcal{R}$ (that are by definition different from the OW-CCA challenge). $\mathcal{M}$ is further capable of successfully simulating a WINV-CMA attacker against the construction (from either StE or CtEaS). Thus $\mathcal{R}$ is expected to return the answer to the OW-CCA challenge, namely the decryption of c. Upon receipt of this answer, $\mathcal{M}$ will forward it to his own challenger. ∎

Lemma 2 *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts a WINV-CMA adversary $\mathcal{A}$ against confirmer signatures from the StE (CtEaS) paradigm to an NM-CPA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that NM-CPA breaks the encryption scheme in question.*

As mentioned previously, this lemma claims that under the assumption of the underlying encryption scheme being NM-CPA secure, there exists no key-preserving reduction $\mathcal{R}$ that reduces NM-CPA breaking the encryption scheme in question to WINV-CMA breaking the construction (from either StE or CtEaS), or if there exists such an algorithm, then the underlying encryption scheme is not NM-CPA secure, thus rendering such a reduction useless.

Proof Let $\mathcal{R}$ be the key-preserving reduction that reduces NM-CPA breaking the encryption scheme underlying the construction to WINV-CMA breaking the construction (from StE or CtEaS) itself. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to NM-CPA break the same encryption scheme by simulating an execution of the WINV-CMA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack w.r.t. a public key pk. $\mathcal{M}$ will launch $\mathcal{R}$ over the same public key pk. Next, $\mathcal{M}$ will simulate a WINV-CMA adversary against the constructions:

- *StE paradigm.* $\mathcal{M}$ (behaving as $\mathcal{A}$) will query $\mathcal{R}$ on two messages m_0, m_1 ($m_0 \neq m_1$) for confirmer signatures. Let μ_0, μ_1 be the corresponding confirmer signatures respectively. $\mathcal{M}$ will further query (μ_i, m_i), $i \in \{0,1\}$, for conversion. Let σ_0, σ_1 be the corresponding answers respectively. We assume that for $i, j \in \{0,1\}$, σ_i is not a valid digital signature (w.r.t. the signer's key) on m_j for $i \neq j$. If this is not the case, $\mathcal{M}$ repeats the experiment until this holds. At that point, $\mathcal{M}$ outputs $D = \{\sigma_0, \sigma_1\}$, to his NM-CPA challenger, as a distribution probability from which the messages will be drawn. He gets in response a challenge encryption $\mu^\star$, of either σ_0 or σ_1 under pk, and is asked to produce a ciphertext μ' whose corresponding plaintext is meaningfully related to the decryption of $\mu^\star$. To solve his task, $\mathcal{M}$ queries for instance $(\mu^\star, m_0)$ for verification. If $\mathcal{R}$ runs the confirmation protocol, i.e. $\mu^\star$ is a valid confirmer signature on m_0, then $\mathcal{M}$ outputs $\Gamma.\mathtt{encrypt}_{pk}(\overline{\sigma_0})$ ($\overline{m}$ refers to the bit-complement of the element m) and the relation R: $R(m, m') = (m' = \overline{m})$. If $\mathcal{R}$ runs the denial protocol, $\mu^\star$ is not a valid confirmer signature on m_0, then $\mathcal{M}$ outputs $\Gamma.\mathtt{encrypt}_{pk}(\overline{\sigma_1})$ and the same relation R. Finally $\mathcal{M}$ aborts the game (stops simulating a WINV-CMA attacker against the generic construction).
- *CtEaS paradigm.* Similarly, $\mathcal{M}$ queries $\mathcal{R}$ on m_0, m_1 ($m_0 \neq m_1$) for signature. Let $\mu_0 = (e_0, c_0, \sigma_0)$ and $\mu_1 = (e_1, c_1, \sigma_1)$ be the corresponding confirmer signatures. $\mathcal{M}$ will then query μ_0, μ_1, along with the corresponding messages, for conversion. Let r_0 and r_1 be the corresponding answers, (i.e., the randomnesses used generate the commitments c_0 and c_1 on m_0 and m_1 resp.). With overwhelming probability, we have $r_0 \neq r_1$ [4], 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 $b \xleftarrow{R} \{0,1\}$ and outputs to his challenger the challenge label $m_b \| \Sigma.pk$, where $\Sigma.pk$ is the public key of the digital signature underlying the construction. $\mathcal{M}$ will receive as a challenge encryption $e^\star$. At that point, $\mathcal{M}$ will query $\mathcal{R}$ on $\mu^\star = (e^\star, c_b, \sigma_b)$ and the message m_b for verification. Note that if $e^\star$ encrypts r_b (under the label $m_b \| \Sigma.pk$), then $\mu^\star$ is valid confirmer signature on m_b, otherwise it is invalid. Therefore, if the outcome of the query is the confirmation protocol, then $\mathcal{M}$ outputs $\Gamma.\mathtt{encrypt}_{pk}(\overline{r_b})$ (w.r.t. the same challenge label $m_b \| \Sigma.pk$), 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, if the outcome of the query is the denial protocol, then $\mathcal{M}$ outputs $\Gamma.\mathtt{encrypt}_{pk}(\overline{r_{1-b}})$ (w.r.t. the same challenge label) and the same relation R. Finally $\mathcal{M}$ aborts the game (stops simulating a WINV-CMA attacker against the construction).

Clearly, $\mathcal{M}$ solves correctly his NM-CPA challenge if $\mathcal{R}$ provides a correct simulation, i.e. $\mathcal{R}$ runs the confirmation protocol for valid signatures, and the denial protocol for invalid ones. According to Fact 1, the advantage of $\mathcal{M}$ is at least equal to the advantage of $\mathcal{R}$. ∎

Theorem 1 *The encryption scheme underlying the above constructions (from either StE or CtEaS) must be at least IND-PCA secure, in case the considered reduction is key-preserving, in order to achieve INV-CMA secure confirmer signatures.*

Proof Corollary 4.1 rules out OW-CPA, OW-PCA, and IND-CPA encryption. Moreover, Lemma 1 rules out OW-CCA encryption. In fact, existence of a key-preserving reduction from the INV-CMA security of the construction to the OW-CCA security of the underlying encryption induces a key-preserving reduction from the WINV-CMA security of the construction to the OW-CCA security of the underlying encryption (reductions between security notions — in this case INV-CMA and WINV-CMA — are key-preserving, and key-preserving reductions are transitive) which contradicts Lemma 1. Similarly, Lemma 2 rules out NM-CPA encryption. The next notion to be considered is then IND-PCA, which concludes the proof. ∎

Remark 6 *The above theorem is only valid when the considered notions are those obtained from pairing a security goal* $\mathsf{goal} \in \{\mathsf{OW}, \mathsf{IND}, \mathsf{NM}\}$ *and an attack model* $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{CCA}\}$. *Presence of other notions will certainly require an additional study.*

Remark 7 *The step of ruling out OW-CPA, OW-PCA, and IND-CPA is necessary although we have proved the insufficiency of stronger notions, namely OW-CCA and NM-CPA. In fact, suppose there is an efficient "useful" key-preserving reduction $\mathcal{R}$ (i.e. $\mathcal{R}$ solves a presumably hard problem) which reduces OW-PCA breaking a*

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

cryptosystem Γ underlying a StE or CtEaS construction to INV-CMA breaking the construction itself. Then there exists an efficient key-preserving reduction say $\mathcal{R}'$ that reduces OW-CCA breaking Γ to INV-CMA breaking the construction (since OW-CCA is stronger than OW-PCA). This does not contradict Lemma 1 as long as Γ is not OW-CCA secure (although it is OW-PCA secure). In other terms, since there are separations between the notions OW-CCA and OW-PCA (same for the other notions), we cannot apply the insufficiency of OW-CCA (NM-CPA) to rule out the weaker notions.

This necessity will become more apparent in Section 8 as we rule out OW-CCA secure encryption in constructions of verifiably encrypted signatures (VES) from StE, yet there exists many realizations of secure VES realizing StE that use OW-PCA encryption (e.g. ElGamal's encryption in bilinear groups) as a building block.

4.3 Extension to arbitrary reductions

To extend the results of the previous paragraph to arbitrary reductions, we first define the notion of *non-malleability of an encryption scheme key generator* through the following two games:
In **Game 0**, we consider an algorithm $\mathcal{R}$ trying to break an encryption scheme Γ, w.r.t. a public key $\Gamma.pk$, 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 encryption scheme Γ. In this game, $\mathcal{R}$ launches $\mathcal{A}$ over his own challenge key $\Gamma.pk$ and some other parameters chosen freely by $\mathcal{R}$ (according to the specifications of $\mathcal{A}$). 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 encryption scheme in question to solving a problem A, which is reducible to OW-CPA breaking the same encryption scheme.
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 $\Gamma.pk' \neq \Gamma.pk$, where $\Gamma.pk$ 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 $\mathsf{succ}_\Gamma^{\mathsf{Game1}}(\mathcal{A}) = \max_{\mathcal{R}} \mathsf{adv}_1(\mathcal{R}^{\mathcal{A}})$ 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 *An encryption scheme 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 negligible in the security parameter.*

This definition informally means that an encryption scheme 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 $pk' \neq pk$.

We generalize now our impossibility results to arbitrary reductions as follows.

Theorem 2 *If the encryption scheme underlying the constructions, from either StE or CtEaS, has a non-malleable key generator, then it must be at least IND-PCA secure in order to achieve INV-CMA secure confirmer signatures.*

To prove this theorem, we first need the following lemma (similar to Lemma 6 of [58])

Lemma 3 *Let $\mathcal{A}$ be an adversary solving a problem A, reducible to OW-CPA breaking an encryption scheme Γ, and let $\mathcal{R}$ be an arbitrary reduction $\mathcal{R}$ that NM-CPA (OW-CCA) breaks an encryption scheme 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 $pk' \neq pk$, $\mathcal{M}$ 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 breaking Γ. Finally, when $\mathcal{R}$ outputs the result to $\mathcal{M}$, the latter will output the same result to his own challenger. ∎

The proof of Theorem 2 is similar to that of Theorem 5 in [58]:

Proof We first remark that the invisibility of the constructions in question is perfectly reducible to OW-CPA breaking the encryption scheme underlying the construction.
Next, we note that the advantage of the meta-reduction $\mathcal{M}$ in the proof of Lemma 2 (Lemma 1) is at least 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 encryption scheme Γ. For instance, this applies to the reduction making the best use of an invisibility adversary $\mathcal{A}$ against the construction. Therefore we have:

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

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

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

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

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

since Δ is negligible, then under the assumption of Γ being NM-CPA (OW-CCA) secure, the advantage of $\mathcal{R}$ is also negligible. ∎

5 Positive Results for CDCS

The above negative results are due to a large extent to the *strong forgeability* of the confirmer signatures from StE or CtEaS. I.e. a polynomial adversary is able to produce, given a valid confirmer signature on a certain message, another valid confirmer signature on the same message without the help of the signer. Indeed, given a valid confirmer signature on a message, an invisibility attacker can request its conversion and then obtain a new confirmer signature on the same message by simply re-encrypting the response (note that a conversion query is not necessary if the used encryption scheme is homomorphic according to Subsection 4.1). Therefore, any reduction $\mathcal{R}$ from the invisibility of the construction to the security of the underlying encryption scheme will need more than a list of records maintaining the queried messages along with the corresponding confirmer and digital signatures. Thus the insufficiency of notions like IND-CPA. In [15, 38, 69], the authors stipulate that the given reduction would need a decryption oracle (of the encryption scheme) in order to handle the queries made by the INV-CMA attacker $\mathcal{A}$, which makes the invisibility of the constructions rest on the IND-CCA security of the encryption scheme. In this section, we remark that the queries made by $\mathcal{A}$ are not completely uncontrolled by $\mathcal{R}$; they are encryptions of some data already released by $\mathcal{R}$, provided the digital signature scheme is strongly unforgeable, and thus known to her. Therefore, a plaintext checking oracle suffices to handle those queries.

The rest of this section will be organized as follows. In Subsection 5.1, we show that StE and CtEaS can thrive on IND-PCA encryption provided the used signature scheme is SEUF-CMA secure. Next, in Subsections 5.2 & 5.3, we propose efficient variants of secure StE and CtEaS respectively which rest on IND-CPA encryption.

5.1 Sufficiency of IND-PCA encryption

Theorem 3 *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0, 1]^2$, confirmer signatures from StE are $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure if the underlying digital signature is (t, ϵ', q_s)-SEUF-CMA secure and the underlying encryption scheme is $(t + q_s q_{sc}(q_{sc} + q_v), \epsilon \cdot (1 - \epsilon')^{(q_{sc}+q_v)}, q_{sc}(q_{sc} + q_v))$-IND-PCA secure.*

Proof Let $\mathcal{A}$ be an attacker that $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA breaks the invisibility of the above confirmer signature, believed to use a (t, ϵ', q_s)-SEUF-CMA secure signature scheme. We will construct an algorithm $\mathcal{R}$ that IND-PCA breaks the underlying encryption scheme as follows:

[keygen] $\mathcal{R}$ gets the public parameters of the target encryption scheme from her challenger, that are $\Gamma.pk$, $\Gamma.\mathtt{encrypt}$, and $\Gamma.\mathtt{decrypt}$. Then, she chooses an appropriate signature scheme Σ with parameters $\Sigma.pk$, $\Sigma.sk$, $\Sigma.\mathtt{sign}$, and $\Sigma.\mathtt{verify}$.

[sign queries] For a signature query on a message m, $\mathcal{R}$ proceeds exactly as the standard algorithm using $\Sigma.sk$ and $\Gamma.pk$. $\mathcal{R}$ will further maintain *internally* in a list $\mathcal{L}$ the queried messages along with the corresponding confirmer signatures and the intermediate values namely the digital signatures (on the message) and the random nonce used to produce the confirmer signatures. It is clear that this simulation is indistinguishable from the standard `sign` algorithm.

[sconfirm queries] $\mathcal{R}$ can execute the standard `sconfirm` protocol on a just generated signature using the randomness used to produce the confirmer signature in question. Again this simulation is indistinguishable from the standard execution of `sconfirm`.

[convert queries] For a putative confirmer signature μ on m, $\mathcal{R}$ will look up the list $\mathcal{L}$. We note that each record of $\mathcal{L}$ comprises three components : (1) the queried message m_i (2) σ_i corresponding to a digital signature on m_i (3) $\Gamma.\mathtt{encrypt}_{\Gamma.pk}(\sigma_i) = \mu_i$, which corresponds to the confirmer signature issued on m_i, and finally (4) r_i corresponding to the randomness used to encrypt σ_i in μ_i. If no record having as first component the message m appears in $\mathcal{L}$, then $\mathcal{R}$ will output $\perp$. Otherwise, let t be the number of records having as first component the message m. $\mathcal{R}$ will invoke the plaintext checking oracle (PCA) furnished by her own challenger on (σ_i, μ), for $1 \leq i \leq t$, where σ_i corresponds to the second component of such records. If the PCA oracle identifies μ as a valid encryption of some σ_i, $1 \leq i \leq t$, then $\mathcal{R}$ will return σ_i, otherwise she will return $\perp$. This simulation differs from the real one when the signature μ is valid and was not obtained from the signing oracle. We note that the only ways to create a valid confirmer signature without the help of $\mathcal{R}$ consist in either encrypting a digital signature obtained from the conversion oracle or coming up with a new fresh pair of message and corresponding signature (m, μ). $\mathcal{R}$ can handle the first case using her PCA oracle and list of records $\mathcal{L}$. In the second case, we can distinguish two sub-cases: either m has not been queried to the signing oracle in which case the pair (m, μ) corresponds to an existential forgery on the confirmer signature scheme and thus to an existential forgery on the underlying digital scheme according to [15, Theorem 1], or m has been queried to the signing oracle but $\Gamma.\mathtt{decrypt}(\mu)$ is not an output of the selective conversion oracle, which corresponds to a strong existential forgery on the underlying digital signature. Therefore, the probability that this scenario does not happen is at least $(1-\epsilon')^{q_{sc}}$ because the underlying digital signature scheme is (t, ϵ', q_s)-SEUF-CMA secure by assumption.

[{confirm, deny} queries] $\mathcal{R}$ will proceed exactly as in the selective conversion with the exception of simulating the denial protocol instead of returning $\perp$, or the confirmation protocol instead of returning the converted digital signature (the {`confirm`, `deny`} protocols are concurrent zero-knowledge proofs of knowledge, and thus they are simulatable). Analogously, the probability that $\mathcal{A}$ does not query a valid signature he has not obtained from the signing oracle is at least $(1-\epsilon')^{q_v}$.

[Challenge phase] Eventually, $\mathcal{A}$ outputs two challenging messages m_0 and m_1. $\mathcal{R}$ will then compute two signatures σ_0 and σ_1 on m_0 and m_1 respectively, which she gives to her own IND-PCA challenger. $\mathcal{R}$ receives then the challenge $\mu^\star$, as the encryption of either σ_0 or σ_1, which she will forward to $\mathcal{A}$.

[Post challenge phase] $\mathcal{A}$ will continue issuing queries to the signing, confirmation/denial and selective conversion oracles and $\mathcal{R}$ can answer as previously. Note that in this phase, $\mathcal{A}$ is not allowed to query the selective conversion or the confirmation/denial oracles on $(m_i, \mu^\star)$, $i = 0, 1$. Also, $\mathcal{R}$ is not allowed to query her PCA oracle on $(\mu^\star, \sigma_i)$, $i = 0, 1$. If during the selective conversion or confirmation/denial queries made by $\mathcal{A}$, $\mathcal{R}$ is compelled to query her PCA oracle on $(\mu^\star, \sigma_i)$, $i = 0, 1$, she will simply output $\perp$ in case of a selective conversion query or simulate the denial protocol in case of a verification query. This differs from the real scenario when $\mu^\star$ is a valid confirmer signature on some message $m \notin \{m_0, m_1\}$, which corresponds to an existential forgery on the underlying signature scheme (σ_i will be a valid digital signature on m_0 or m_1 and on a message $m \notin \{m_0, m_1\}$). Again, this does not happen with probability at least $(1-\epsilon')^{q_{sc}+q_v}$.

[Final output] When $\mathcal{A}$ outputs his answer $b \in \{0, 1\}$, $\mathcal{R}$ will forward this very answer to her own challenger. Therefore $\mathcal{R}$ will IND-PCA break the underlying encryption scheme with advantage at least $\epsilon \cdot (1-\epsilon')^{(q_v+q_{sc})}$, in time at most $t + q_s q_{sc}(q_v + q_{sc})$ after at most $q_{sc}(q_{sc} + q_v)$ queries to the PCA oracle.

For the CtEaS paradigm, we first need this lemma.

Lemma 4 *Let Ω and Γ be a commitment and a public key encryption scheme respectively. We consider the following game between an adversary $\mathcal{A}$ and his challenger $\mathcal{R}$:*

1. *$\mathcal{R}$ invokes the algorithms $\Gamma.\mathtt{keygen}(1^\kappa)$ to generate (pk, sk), where κ is a security parameter.*
2. *$\mathcal{A}$ outputs two messages m_0 and m_1 such that $m_0 \neq m_1$ to his challenger.*
3. *$\mathcal{R}$ generates two nonces r_0 and r_1 such that $r_0 \neq r_1$. Next, he chooses two bits $b, b' \xleftarrow{R} \{0,1\}$ uniformly at random. Finally, he outputs to $\mathcal{A}$ $c_b = \Omega.\mathtt{commit}(m_b, r_{1-b'})$ and $e_{b'} = \Gamma.\mathtt{encrypt}_{pk}(r_{b'})$.*
4. *$\mathcal{A}$ outputs a bit b_a representing his guess of c_b not being the commitment of m_b using the nonce $\Gamma.\mathtt{decrypt}(e_{b'})$. $\mathcal{A}$ wins the game if $b_a \neq b$, and we define his advantage to be*

$$\mathsf{adv}(\mathcal{A}) = \left|\Pr[b \neq b_a] - \frac{1}{2}\right|,$$

where the probability is taken over the random tosses of both $\mathcal{A}$ and $\mathcal{R}$.

If Ω is injective, binding, and (t_h, ϵ_h)-hiding, then $\mathsf{adv}(\mathcal{A})$ in the above game is at most ϵ_h.

Proof Let ϵ be the advantage of $\mathcal{A}$ in game above. We will construct an adversary $\mathcal{R}$ which breaks the hiding property of the used commitment with advantage ϵ.

- $\mathcal{R}$ gets from $\mathcal{A}$ the message m_0, m_1, and forwards them to her own challenger.
- $\mathcal{R}$ receives from her challenger the commitment $c_b = \Omega.\mathtt{commit}(m_b, r)$ for some $b \xleftarrow{R} \{0,1\}$ and some nonce r.
- $\mathcal{R}$ generates a nonce r' and outputs to $\mathcal{A}$ c_b and $e = \Gamma.\mathtt{encrypt}_{pk}(r')$.
- When $\mathcal{A}$ outputs a bit b_a, $\mathcal{R}$ outputs to her challenger $1 - b_a$.

If $\mathcal{A}$ can by some means get hold of r', then he can compute $c_i = \Omega.\mathtt{commit}(m_i, r')$, $i = 0, 1$. Since Ω is injective and binding then $c_b \neq \Omega.\mathtt{commit}(m_b, r')$ and $c_b \neq \Omega.\mathtt{commit}(m_{1-b}, r')$ respectively, i.e. $c_b \notin \{c_0, c_1\}$. Thus, $\mathcal{A}$ will get no information on the message underlying c_b even if he manages to invert e.

We have by definition:

$$\begin{aligned} \epsilon_h = \mathsf{adv}(\mathcal{R}) &= \left|\Pr[1 - b_a = b] - \frac{1}{2}\right| \\ &= \left|\Pr[b_a \neq b] - \frac{1}{2}\right| \\ &= \mathsf{adv}(\mathcal{A}) \end{aligned}$$

∎

Remark 8 *Note that the above lemma holds true regardless of the used encryption Γ. For instance, it can be used with encryption schemes which support labels and which do not require any kind of security.*

Theorem 4 *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, confirmer signatures from CtEaS are $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure if they use a (t, ϵ', q_s)-SEUF-CMA secure digital signature, an injective, binding and (t, ϵ_h)-hiding commitment, and a $(t + q_s q_{sc}(q_{sc} + q_v), \frac{1}{2}(\epsilon + \epsilon_h) \cdot (1 - \epsilon')^{(q_{sc}+q_v)}, q_{sc}(q_{sc} + q_v))$-IND-PCA secure encryption scheme with labels.*

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

[`setup` and `keygen`]. $\mathcal{R}$ gets the parameters of the encryption scheme Γ from her challenger. Then she chooses a (t, ϵ', q_s)-SEUF-CMA secure digital signature Σ (along with a key pair $(\Sigma.pk, \Sigma.sk)$) and an injective, binding and (t, ϵ_h)-hiding commitment Ω. $\mathcal{R}$ sets the above entities as components of the construction $\mathcal{A}$ is trying to attack.

[`sign` and `sconfirm` queries]. $\mathcal{R}$ proceeds exactly as the standard algorithm or protocol, with the exception of maintaining in a list $\mathcal{L}$ the queried messages, the corresponding confirmer signatures, and the intermediate values used to produce these, for instance the random string used to produce the commitment on the message in question.
Clearly, $\mathcal{R}$ can perfectly simulate the standard `sign` and `sconfirm` procedures.

[`convert` and {`confirm`, `deny`} queries] To convert an alleged signature $\mu_i = (e_i, c_i, \sigma_i)$ on a message m_i, $\mathcal{R}$ checks the validity of σ_i on c_i; if it is invalid, then $\mathcal{R}$ proceeds as the standard algorithm. Otherwise, $\mathcal{R}$ checks the list $\mathcal{L}$ for records corresponding to the queried message m_i and where c_i has been used also as a commitment on the message (to generate the corresponding signature). If e_i is found in one of these

records as encryption of the opening value of c_i, then $\mathcal{R}$ proceeds as the standard algorithm. Otherwise, $\mathcal{R}$ queries her PCA oracle on e_i and on the opening value of c_i found in one of these records, if the answer is 'yes', then $\mathcal{R}$ returns this very opening value, otherwise $\mathcal{R}$ returns $\perp$. Verification ($\{\texttt{confirm}, \texttt{deny}\}$) queries are handled similarly with the exception of simulating the denial protocol instead of $\perp$, and the confirmation protocol instead of converting the signature (the confirmation/denial protocols are simulatable as they are concurrent zero-knowledge proofs of knowledge).
This simulation differs from the standard algorithm when μ_i is valid, but m_i has not been queried before, or c_i has not been used to generate commitments on m_i. The first case corresponds to an existential forgery on the construction which results in an existential forgery on the underlying digital signature scheme. The second case corresponds to an existential forgery on the underlying signature scheme. Both cases do not happen with probability at least $(1-\epsilon')^{q_v+q_{sc}}$.

[Challenge phase] At some point, $\mathcal{A}$ will output two messages m_0, m_1 to $\mathcal{R}$. 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 her challenger the label $m_b \| \Sigma.pk$ and the strings r_0, r_1. She receives then a ciphertext $e_{b'}$, encryption of $r_{b'}$, for some $b' \xleftarrow{R} \{0,1\}$. To answer her 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 $\mu = (e_{b'}, c_b, \Sigma.\texttt{sign}_{\Sigma.sk}(c_b))$ as a challenge signature to $\mathcal{A}$. Two cases: either μ is a valid confirmer signature on m_b (if $b' = b''$), or it is not a valid signature on neither m_0 nor m_1. If the advantage of $\mathcal{A}$ is non-negligibly different from the advantage of an INV-CMA attacker in a real attack, then , according to Lemma 4, $\mathcal{A}$ can be used to break the hiding property of Ω.

[Post challenge phase] $\mathcal{A}$ continues to issue queries and $\mathcal{R}$ continues to handle them as before. Note that at this stage, $\mathcal{R}$ cannot request her PCA oracle on $(e_{b'}, r_i)$, $i \in \{0,1\}$ under the label $m_b \| \Sigma.pk$. $\mathcal{R}$ would need to make such a query if she gets a verification (conversion) query on a signature $(e'_b, c_b, -) \neq \mu$ and the message m_b. $\mathcal{R}$ will respond to such a query by running the denial protocol (output $\perp$). This simulation differs from the real algorithm when $(e'_b, c_b, -)$ 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 a strong existential forgery on the digital signature scheme underlying the construction.

[Final output] The rest of the proof follows in a straightforward way. Now, let $\mu = (e_{b'}, c_b, \Sigma.\texttt{sign}_{\Sigma.sk}(c_b))$ be the challenge signature. Let b_a be the bit output by $\mathcal{A}$. $\mathcal{R}$ will output b'' to her challenger in case $b = b_a$ and $1 - b''$ otherwise.
The advantage of $\mathcal{A}$ in such an attack is defined by

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

We also have

$$\epsilon_h = \left| \Pr[b_a \neq b | b' \neq b''] - \frac{1}{2} \right|$$

We assume again without loss of generality that $\epsilon = \Pr[b_a = b|b' = b''] - \frac{1}{2}$ and $e_h = \Pr[b_a \neq b|b' \neq b''] - \frac{1}{2}$. The advantage of $\mathcal{R}$ is then given by

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

The last but one equation is due to the facts $\Pr[b' \neq b''] = \Pr[b' = b''] = \frac{1}{2}$ as $b'' \xleftarrow{R} \{0,1\}$, and to the fact that, in case $b' \neq b''$, the probability that $\mathcal{A}$ answers $1 - b$ is $\frac{1}{2}$ greater than the advantage of the adversary in the game defined in Lemma 4, which is equal to ϵ_h.

5.2 An efficient variant of StE

One attempt to circumvent the problem of *strong forgeability* of constructions obtained from the plain StE paradigm can be achieved by binding the digital signature to its encryption. In this way, from a digital signature σ and a message m, an adversary cannot create a new confirmer signature on m by just re-encrypting σ. In fact, σ forms a digital signature on m and some data, say c, which uniquely defines the confirmer signature on m. Moreover, this data c has to be public in order to issue the $\{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}$ protocols.

In this subsection, we propose a realization of this idea using hybrid encryption (the KEM/DEM paradigm). We also allow more flexibility without compromising the overall security by encrypting only one part of the signature and leaving out the other part, provided it does not reveal information about the key nor about the message.

The construction Let Σ be a digital signature scheme given by $\Sigma.\mathtt{keygen}$, which generates a key pair ($\Sigma.sk$, $\Sigma.pk$), $\Sigma.\mathtt{sign}$, and $\Sigma.\mathtt{verify}$. Let furthermore $\mathcal{K}$ be a KEM given by $\mathcal{K}.\mathtt{keygen}$, which generates a key pair ($\mathcal{K}.pk$, $\mathcal{K}.sk$), $\mathcal{K}.\mathtt{encap}$, and $\mathcal{K}.\mathtt{decap}$. Finally, we consider a DEM $\mathcal{D}$ given by $\mathcal{D}.\mathtt{encrypt}$ and $\mathcal{D}.\mathtt{decrypt}$.

We assume that any digital signature σ, generated using Σ on an arbitrary message m, can be efficiently transformed in a reversible way to a pair (s, r) where r reveals no information about m nor about $(\Sigma.sk, \Sigma.pk)$. I.e. there exists an algorithm that inputs a message m and a key pair $(\Sigma.sk, \Sigma.pk)$ and outputs a string statistically indistinguishable from r, where the probability is taken over the messages and the key pairs considered by Σ. This technical detail will improve the efficiency of the construction as it will not necessitate encrypting the entire signature σ, but only the message-key-dependent part, namely s. Finally, we assume that s belongs to the message space of $\mathcal{D}$.

We further assume that the encapsulations generated by $\mathcal{K}$ are exactly κ-bit long, where κ is a security parameter. This can be for example realized by padding with zeros, on the left of the most significant bit of the given encapsulation, until the resulting string has length κ. Moreover, the operator $\|$ denotes the usual concatenation operation between two bit-strings. As a result, the first bit of m will always be at the $(\kappa + 1)$-st position in $c\|m$, where c is a given encapsulation. Such a technical detail will play an important role in the unforgeability and invisibility of the construction.

The construction of confirmer signatures from Σ, $\mathcal{K}$, and $\mathcal{D}$ is given as follows.

Key generation (keygen). Set the signer's key pair to $(\Sigma.sk, \Sigma.pk)$ and the confirmer's key pair to $(\mathcal{K}.sk, \mathcal{K}.pk)$.

Signature (sconfirm). Fix a key k together with its encapsulation c. Then, compute a (digital) signature $\sigma = \Sigma.\mathtt{sign}_{\Sigma.sk}(c\|m) = (s, r)$ on $c\|m$, and output $\mu = (c, \mathcal{D}.\mathtt{encrypt}_k(s), r)$.

Verification ({sconfirm, confirm, deny}). To confirm (deny) a purported signature $\mu = (c, e, r)$, issued on a certain message m, the prover (either the signer or the confirmer) provide a concurrent zero-knowledge proof of knowledge of the decryption of (c, e), which together with r forms a valid (invalid) signature on $c\|m$. Providing such a proof is possible since the underlying statement is an NP language [28].

Selective conversion (convert). To convert a signature $\mu = (c, e, r)$ issued on a certain message m, the confirmer first checks its validity. If it is invalid, he outputs $\perp$, otherwise, he computes $k = \mathcal{K}.\mathtt{decap}_{\mathcal{K}.sk}(c)$ and outputs $(\mathcal{D}.\mathtt{decrypt}_k(e), r)$.

Theorem 5 *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, the above construction is (t, ϵ, q_s)-EUF-CMA secure if the underlying digital signature scheme is (t, ϵ, q_s)-EUF-CMA secure.*

Proof Let $\mathcal{A}$ be an attacker that (t, ϵ, q_s)-EUF-CMA breaks the existential unforgeability of the above construction. We will construct an adversary $\mathcal{R}$ that (t, ϵ, q_s)-EUF-CMA breaks the underlying digital signature scheme:

[Key generation] $\mathcal{R}$ gets the parameters of the signature scheme in question from her challenger. Then she chooses an appropriate KEM $\mathcal{K}$ and DEM $\mathcal{D}$ and asks $\mathcal{A}$ to provide her with the confirmer key pair $(\mathcal{K}.sk, \mathcal{K}.pk)$. Finally, $\mathcal{R}$ fixes the above parameters as a setting for the confirmer signature scheme $\mathcal{A}$ is trying to attack.

[Signature queries] For a signature query on a message m, $\mathcal{R}$ will first compute an encapsulation c together with its decapsulation k (using $\Gamma.pk$). Then, she will request her challenger for a digital signature $\sigma = (s, r)$ on $c\|m$. Finally, she encrypts s in $\mathcal{D}.\mathtt{encrypt}_k(s)$, then outputs the confirmer signature $(c, \mathcal{D}.\mathtt{encrypt}_k(s), r)$.

Experiment $\mathbf{Exp}^{\text{invisibility}}(1^\kappa)$

1. $param \leftarrow \texttt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \texttt{keygen}_\mathsf{S}(1^\kappa)$; $(pk_\mathsf{C}, sk_\mathsf{C}) \leftarrow \texttt{keygen}_\mathsf{C}(1^\kappa)$;
3. $(m^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\text{find}, pk_\mathsf{S}, pk_\mathsf{C})$
   $\mathfrak{S} : m \longmapsto \texttt{sign}_{\{sk_\mathsf{S}, pk_\mathsf{C}\}}(m)$
   $\mathfrak{Cv} : (m, \mu) \longmapsto \texttt{convert}_{sk_\mathsf{C}}(m, \mu)$
   $\mathfrak{V} : (m, \mu) \longmapsto \{\texttt{sconfirm}, \texttt{confirm}, \texttt{deny}\}(m, \mu, pk_\mathsf{C}, pk_\mathsf{S})$
4. $\mu_1^\star \leftarrow \texttt{sign}\{sk_\mathsf{S}, pk_\mathsf{S}, pk_\mathsf{C}\}(m^\star)$
5. $\mu_0^\star \overset{R}{\leftarrow} \mathsf{CS}$; $b \overset{R}{\leftarrow} \{0,1\}$
6. $b^\star \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\text{guess}, \mathcal{I}, \mu_b^\star, pk_\mathsf{S}, pk_\mathsf{C})$
   $\mathfrak{S} : m \longmapsto \texttt{sign}\{sk_\mathsf{S}, pk_\mathsf{C}\}(m)$
   $\mathfrak{Cv} : (m, \mu)(\neq (m^\star, \mu_b^\star)) \longmapsto \texttt{convert}_{sk_\mathsf{C}}(m, \mu)$
   $\mathfrak{V} : (m, \mu)(\neq (m^\star, \mu_b^\star)) \longmapsto \{\texttt{sconfirm}, \texttt{confirm}, \texttt{deny}\}(m, \mu, pk_\mathsf{C}, pk_\mathsf{S})$
7. If $(b = b^\star)$ return 1 else return 0.

[Final Output] Once $\mathcal{A}$ outputs his forgery $\mu^\star = (c^\star, e^\star, r^\star)$ on $m^\star$. $\mathcal{R}$ will compute the decapsulation of $c^\star$, say k. If $\mu^\star$ is valid then by definition $(\mathcal{D}.\texttt{decrypt}_k(e^\star), r^\star)$ is a valid digital signature on $c^\star \| m^\star$. Thus, $\mathcal{R}$ outputs $(\mathcal{D}.\texttt{decrypt}_k(e^\star), r^\star)$ and $c^\star \| m^\star$ as a valid existential forgery on Σ. In fact, if, during a query made by $\mathcal{A}$ on a message m^i, $\mathcal{R}$ is compelled to query her own challenger for a digital signature on $c^\star \| m^\star = c^i \| m^i$, then $m^\star = m^i$ (by construction), which contradicts the fact that $(\mu^\star, m^\star)$ is an existential forgery output by $\mathcal{A}$.

Note that there will be no need to simulate the confirmation/denial and selective conversion oracles since $\mathcal{A}$ knows $\mathcal{K}.sk$ which allows the verification of the confirmer signatures. ∎

The following remark is vital for the invisibility of the resulting confirmer signatures.

Remark 9 *The previous theorem shows that existential unforgeability of the underlying digital signature scheme suffices to ensure existential unforgeability of the resulting construction. Actually, one can also show that this requirement on the digital signature (EUF-CMA security) guarantees that no adversary, against the construction, can come up with a valid confirmer signature $\mu = (c, e, r)$ (c is the encapsulation used to generate the confirmer signature μ) on a message m that has been queried before to the signing oracle but where c was never used to generate answers (confirmer signatures) to the signature queries.*

To prove this claim, we construct from such an adversary, say $\mathcal{A}$, an EUF-CMA adversary $\mathcal{R}$ against the underlying digital signature scheme, which runs in the same time and has the same advantage as $\mathcal{A}$. In fact, $\mathcal{R}$ will simulate $\mathcal{A}$'s environment in the same way described in the proof of Theorem 5. When $\mathcal{A}$ outputs his forgery $\mu^\star = (c^\star, e^\star, r^\star)$ on a message m_i that has been previously queried to the signing oracle, $\mathcal{R}$ decrypts $(c^\star, e^\star)$ in $s^\star$, which by definition forms, together with $r^\star$, a valid digital signature on $c^\star \| m_i$. Since by assumption $c^\star$ was never used to generate confirmer signatures on the queried messages, $\mathcal{R}$ never invoked her own challenger for a digital signature on $c^\star \| m_i$. Therefore, $(s^\star, r^\star)$ will form a valid existential forgery on the underlying digital signature scheme.

In the rest of this subsection, we show that the new StE paradigm achieves a stronger notion (than INV-CMA) of invisibility that we denote SINV-CMA. This notion was first introduced in [36], and it captures the difficulty to distinguish confirmer signatures on an adversarially chosen message from random elements in the confirmer signature space. Again, the difference with the previously mentioned notions lies in the challenge phase where the SINV-CMA attacker outputs a message $m^\star$ and receives in return an element $\mu^\star$ which is either a confirmer signature on $m^\star$ or a random element from the confirmer signature space. There is again the natural restriction of not querying the challenge pair to the `sconfirm`, {`confirm`, `deny`}, and `convert` oracles.

Theorem 6 *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, the construction proposed above is $(t, \epsilon, q_s, q_v, q_{sc})$-SINV-CMA secure if it uses a (t, ϵ', q_s)-EUF-CMA secure digital signature, an INV-OT secure DEM and an $(t + q_s(q_v + q_{sc}), \epsilon \cdot (1 - \epsilon')^{q_v + q_{sc}})$-IND-CPA secure KEM.*

Proof Let $\mathcal{A}$ be an attacker that $(t, \epsilon, q_s, q_v, q_{sc})$-SINV-CMA breaks our construction, assumed to use a (t, ϵ', q_s)-EUF-CMA secure digital signature and an INV-OT secure DEM. We will construct an algorithm $\mathcal{R}$ that $(t + q_s(q_v + q_{sc}), \epsilon \cdot (1 - \epsilon')^{q_v + q_{sc}})$-IND-CPA breaks the underlying KEM.

[`keygen`**]** $\mathcal{R}$ gets the parameters of the KEM $\mathcal{K}$ from his challenger. Then, she chooses an appropriate INV-OT secure DEM $\mathcal{D}$ together with an (t, ϵ', q_s)-EUF-CMA secure signature scheme Σ. $\mathcal{R}$ further generates a key pair $(\Sigma.sk, \Sigma.pk)$for Σ and sets it as the signer's key pair.

[`sign` **and** `sconfirm` **queries]** For a signature query, $\mathcal{R}$ proceeds as the standard algorithm. She further maintains a list $\mathcal{L}$ of the encapsulations c and keys k used to generate the confirmer signatures.

[$\{$`confirm`, `deny`$\}$ **queries]** For a verification query on a signature $\mu = (c, e, r)$ and a message m, R will look up the list $\mathcal{L}$ for the decapsulation of c, which once found allows $\mathcal{R}$ to check the validity of the signature and therefore simulate correctly the suitable protocol (confirmation or denial). If c has not been used to generate confirmer signatures, then $\mathcal{R}$ will run the denial protocol. Note that $\mathcal{R}$ is able to perfectly simulate to confirmation/denial protocol since these protocols are by definition concurrent zero-knowledge proofs of knowledge.
This simulation differs from the real one when the signature $\mu = (c, e, r)$ on m is valid, but c does not appear in any record of $\mathcal{L}$. We distinguish two cases: either m was never queried to the signing oracle, then (m, μ) would correspond to an existential forgery on the confirmer signature (and thus to an existential forgery on Σ), or m has been previously queried to the signing oracle in which case (m, μ) would correspond to an existential forgery on Σ thanks to Remark 9. Hence, the probability that both scenarios do not happen is at least $(1 - \epsilon')^{q_v}$ (Σ is (t, ϵ', q_s)-EUF-CMA secure).

[`convert` **queries]** Conversion queries are treated like verification queries with the exception of converting the signature instead of running `confirm`, and issuing $\perp$ instead of running `deny`. Similarly, this simulation does not differ from the real execution of the algorithm with probability at least $(1 - \epsilon')^{q_{sc}}$.

[Challenge] Eventually, $\mathcal{A}$ outputs a challenge message $m^\star$. $\mathcal{R}$ uses her challenge $(c^\star, k^\star)$ to compute a digital signature $(s^\star, r^\star)$ on $c^\star \| m^\star$. Then, she encrypts $s^\star$ using $k^\star$ and outputs $\mu^\star = (c^\star, \mathcal{D}.\texttt{encrypt}_{k^\star}(s^\star), r^\star)$ to $\mathcal{A}$. Therefore, $\mu^\star$ is either a valid confirmer signature on $m^\star$ or an element indistinguishable from a random element in the confirmer signature space ($k^\star$ is random and $\mathcal{D}$ is INV-OT secure, moreover $r^\star$ is information-theoretically independent from m and $\Sigma.pk$). If $\mu^\star$, in the latter case, is a random element in the confirmer signature space, then this complies with the scenario of a real attack. Otherwise, if $\mu^\star$ is *only indistinguishable from random*, then if the advantage of $\mathcal{A}$ is non-negligibly different from the advantage of an invisibility adversary in a real attack, then $\mathcal{A}$ can be easily turned into an attacker against the INV-OT security property of $\mathcal{D}$. To sum up, under the INV-OT assumption of $\mathcal{D}$, the challenge confirmer signature $\mu^\star$ is either a valid confirmer signature on $m^\star$ or a random element in the confirmer signature space.

[Post challenge phase] $\mathcal{A}$ will continue issuing queries to the previous oracles, and $\mathcal{R}$ can answer as previously. Note that in this phase, $\mathcal{A}$ might request the verification or selective conversion of a confirmer signature $(c^\star, -, -)$ on a message m_i. In this case, $\mathcal{R}$ will simply issue the denial protocol in case of a verification query, or the symbol $\perp$ in case of a conversion query. Following the same analysis as above, the probability that the simulation does not differ from the real execution is at least $(1-\epsilon')^{q_{sc}+q_v}$.

[Final output] When $\mathcal{A}$ outputs his answer $b \in \{0, 1\}$, $\mathcal{R}$ will forward this answer to her own challenger. Therefore $\mathcal{R}$ will $(t + q_s(q_v + q_{sc}), \epsilon \cdot (1 - \epsilon')^{q_v + q_{sc}})$-IND-CPA break $\mathcal{K}$.

Note that the strong unforgeability of the underlying signature scheme is not needed here to achieve invisibility. In fact, if the adversary can come up with another digital signature (s', r') on a given $c\|m$, then there is just one way to create the corresponding confirmer signature, namely encrypt s using $k = \mathcal{K}.\texttt{decap}(c)$. Therefore, the reduction is able to handle a query requesting the confirmation/denial or selective conversion of such a signature by just maintaining a list of the used encapsulations c and the corresponding decapsulations k.

5.3 An efficient variant of CtEaS

We remeditate to the strong forgeability problem of CtEaS by using the same trick applied to StE, namely bind the used digital signature to the corresponding confirmer signature. This is achieved by producing a digital signature on both the commitment and the encryption of the random string used to generate it. In this way, the attack discussed in Subsection 4.1 no longer applies, since an adversary will need to produce a digital signature on the commitment and the re-encryption of the random string used in it.

Note that such a fix already appears in the construction of [38], however, it was not exploitable as the invisibility was considered in the insider model.

5.3.1 The construction

Let Σ be a signature scheme given by $\Sigma.\mathtt{keygen}$, that generates $(\Sigma.pk, \Sigma.sk)$, $\Sigma.\mathtt{sign}$, and $\Sigma.\mathtt{verify}$. Let further Γ denote an encryption scheme given by $\Gamma.\mathtt{keygen}$, that generates $(\Gamma.pk, \Gamma.sk)$, $\Gamma.\mathtt{encrypt}$, and $\Gamma.\mathtt{decrypt}$. We note that Γ does need to support labels in our construction. Finally let Ω denote a commitment scheme given by $\Omega.\mathtt{commit}$ and $\Omega.\mathtt{open}$. We assume that Γ produces ciphertexts of length exactly a certain κ. As a result, the first bit of c will always be at the $(\kappa+1)$-st position in $e\|c$, where e is an encryption produced by Γ. The signer's key pair is $(\Sigma.pk, \Sigma.sk)$ and the confirmer's key pair is $(\Gamma.pk, \Gamma.sk)$. To sign a message m, the signer produces a commitment c on m using a random string r, encrypts this string in e, and then produces a digital signature $\sigma = \Sigma.\mathtt{sign}_{\Sigma.sk}(e\|c)$. Finally, the signer outputs $\mu = (e, c, \sigma)$ as a confirmer signature on m. To verify ($\{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}$) a signature on a given message, the prover (either the signer or the confirmer) provides a concurrent zero-knowledge proof of the statement in question. Finally, conversion is achieved by revealing the decryption of e.

It clear that this new construction looses the parallelism of the original one, i.e. encryption and signature can no longer be carried out in parallel, however, it has the advantage of resting on cheaper encryption as we will show in the following.

Theorem 7 *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]^2$, the construction depicted above is (t, ϵ, q_s)-EUF-CMA secure if it uses a statistically binding commitment scheme and a (t, ϵ, q_s)-EUF-CMA secure digital signature scheme.*

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 her attacker, and chooses a suitable encryption and commitment scheme. After she gets the confirmer's key pair from $\mathcal{A}$, $\mathcal{R}$ can perfectly simulate signature queries using her own challenger. At some point, $\mathcal{A}$ will output a forgery $\mu^\star = (e^\star, c^\star, \sigma^\star)$ on some message $m^\star$, which was never queried before for signature. By definition, $\sigma^\star$ is a valid digital signature on $e^\star\|c^\star$. It will form an existential forgery on the digital signature scheme if $e^\star\|c^\star$ was never queried before by $\mathcal{R}$ for a digital signature. Suppose there exists $1 \le i \le q_s$ such that $e^\star\|c^\star = e_i\|c_i$ where $\mu_i = (e_i, c_i, \sigma_i)$ was the output confirmer signature on the query m_i. Due to the special way the strings $e_i\|c_i$ are created, equality of the strings $e^\star\|c^\star$ and $e_i\|c_i$ implies equality of their suffixes (that start at the $(\kappa+1)$-st position), namely $c^\star$ and c_i. This equality implies the equality of m_i and $m^\star$ since the used commitment is binding by assumption. Thus, $\mathcal{R}$ returns $(\sigma^\star, e^\star\|c^\star)$ as a valid existential forgery against the digital signature in question. ∎

Theorem 8 *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, the construction depicted above is $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure if it uses an (t, ϵ', q_s)-SEUF-CMA secure digital signature, an injective, statistically binding, and (t, ϵ_h)-hiding commitment, and a $(t + q_s(q_v + q_{sc}), \frac{1}{2}(\epsilon + \epsilon_h)(1-\epsilon')^{q_v+q_{sc}})$-IND-CPA secure encryption scheme.*

Proof The proof is very similar to that of Theorem 4. We therefore avoid the details and focus on the changes. Let $\mathcal{R}$ be the reduction using the attacker $\mathcal{A}$ against the construction in order to break the underlying encryption. Simulation of `sign` and `sconfirm` queries is done as in the mentioned proof. For instance, $\mathcal{R}$ maintains a list $\mathcal{L}$ of the strings used to produce commitments on the queried messages in addition to their encryptions. For a verification (conversion) query, $\mathcal{R}$ looks up the list $\mathcal{L}$ for the decryption of the first component of the signature; if it is found, $\mathcal{R}$ simulates the confirmation protocol (issues the converted signature in case of a conversion query), otherwise she simulates the denial protocol (issues the symbol $\perp$ in case of a conversion). The difference between this simulation and the real execution of the algorithm manifest when a queried signature, say (e_i, c_i, σ_i), is valid, on the queried message m_i, but e_i is not present in the list. 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, let (e_j, c_j, σ_j) be the output signature to $\mathcal{A}$ on the message m_i. We have $e_i\|c_i \neq e_j\|c_j$ since $e_i \neq e_j$, and both e_i and e_j are the n-bit prefixes of $e_i\|c_i$ and $e_j\|c_j$ resp. We conclude that the adversary would have to compute a digital signature on a string for which he never had obtained a signature. Thus, 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 $(1-\epsilon')^{q_v+q_{sc}}$. The rest is similar to the proof of Theorem 4. ∎

Prover P		Verifier V
$c_1 = \mathtt{Cmt}_1(x)$		
$(c_2, b_2, z_2) = \mathtt{trSim}_2(pk_\mathsf{V})$	$\xrightarrow{(c_1, c_2)}$	
	$\xleftarrow{\ b\ }$	Choose $b \xleftarrow{R} \{0,1\}^\ell$ $(b \in \mathbb{N})$
$b_1 = b - b_2$		
$z_1 = \mathtt{Rsp}_1(c_1, w, b_1)$	$\xrightarrow{(z_1, z_2, b_1, b_2)}$	$d_0 = (b = b_1 + b_2)$
		$d_1 = \mathtt{Dcd}_1(x, c_1, b_1, z_1)$
		$d_2 = \mathtt{Dcd}_2(pk_\mathsf{V}, c_2, b_2, z_2)$
		return $d_0 \wedge d_1 \wedge d_2$

Figure 2: Proof of membership to the language $\{(x,w) \in \mathcal{R} \vee (sk_\mathsf{V}, pk_\mathsf{V}) \in \mathsf{Valid}\}$ Common input : (x, pk_V) and Private input : w or sk_V

Remark 10 *Both Theorem 7 and Theorem 8 can be used with computationally binding commitments. The only issue is to have the formulation of both theorems complicated by further terms, e.g. ϵ_b, if we use a (t, ϵ_b)-binding commitment.*

6 Practical Realizations of CDCS

In this section, we present practical realizations of confirmer signatures from StE, CtEaS, and EtS. We first start by showing how to achieve online non-transferability from offline one if the confirmation/denial protocols are Σ protocols. Then, we describe some classes of basic primitives that constitute important building blocks for the mentioned constructions. Finally, we proceed to the description of some practical realizations of confirmer signatures from StE, CtEaS, and EtS.

6.1 Online versus offline non-transferability

Online non-transferability as previously mentioned allows to avoid some attacks in which the intended verifier, say V, interacts *concurrently* with the genuine prover and a hidden malicious verifier $\widetilde{V}$ such that this latter gets convinced of the proven statement (validity or invalidity of a signature w.r.t. a given message). One way to circumvent this problem consists in using designated verifier proofs [47]. The idea is that such proofs can be constructed by both the prover and the verifier. When a verifier receives such a proof, he will be convinced of the validity of the underlying statement since he has not constructed it himself. However, he cannot convince a third party of the validity of the statement as he can himself perfectly simulate the answers sent by the prover.

A generic construction of designated verifier proofs from Σ protocols was given in [66]. The idea consists in proving either the statement in question or proving knowledge of the verifier's private key. This is achieved using *proofs of disjunctive knowledge* if the proof of the statement in question and the proof of knowledge of the verifier's private key are both Σ protocols. More precisely, let the statement to be proven be of the form $(x,w) \in \mathcal{R}$ where $\mathcal{R}$ is some NP relation, and (x,w) is the pair (instance,relation). Let further $(sk_\mathsf{V}, pk_\mathsf{V})$ be the verifier's key pair. Suppose that both the statements $(x,w) \in \mathcal{R}$ and $(sk_\mathsf{V}, pk_\mathsf{V}) \in \mathsf{Valid}$ can be proven using Σ protocols. The work-flow of the proof:

$$\mathsf{PoK}\{(x,w) \in \mathcal{R} \vee (sk_\mathsf{V}, pk_\mathsf{V}) \in \mathsf{Valid}\}$$

is depicted in Figure 2.

It is clear that this protocol can be also carried out using the private the input sk_V. We refer to [66] for further details about the analysis of this protocol.

To sum up, if $\{\mathtt{sconfirm}, \mathtt{confirm}, \mathtt{deny}\}$ are Σ protocols, then they can be efficiently transformed into designated verifier proofs of knowledge providing therefore the online non-transferability for the signatures. Note that if we consider such proofs in the registrated key model, i.e. a model that requires the verifier to prove knowledge of his private key before a certifying authority, the reduction can easily run these protocols for the invisibility adversary given the knowledge of the verifier's private key.

Prover P		Verifier V
Choose $s' \xleftarrow{R} \mathbb{G}$		
Compute $t_1 = f(s')$	$\xrightarrow{\quad t_1 \quad}$	
	$\xleftarrow{\quad b \quad}$	Choose $b \xleftarrow{R} \{0,1\}^{\ell}$ $(b \in \mathbb{N})$
	$\xrightarrow{z = s' * s^b}$	
		Verify that $f(z) = t_1 I^b$

Figure 3: Proof system for membership to the language $\{s\colon f(s) = I\}$ Common input: I and Private input : s

6.2 The class $\mathbb{S}$ of signatures

Definition 4 *$\mathbb{S}$ is the set of all digital signatures for which there exists a pair of efficient algorithms,* `convert` *and* `retrieve`*, where* `convert` *inputs a public key pk, a message m, and a valid signature σ on m (according to pk) and outputs the pair (s, r) such that:*

1. *r reveals no information about m nor about pk, i.e. there exists an algorithm* simulate *such that for every public key pk from the key space and for every message m from the message space, the output* simulate(pk, m) *is statistically indistinguishable from r.*
2. *there exists an algorithm* `compute` *that on the input pk, the message m and r, computes a description of a function $f : (\mathbb{G}, *) \to (\mathbb{H}, \circ_s)$:*
   - *where $(\mathbb{G}, *)$ is a group and $\mathbb{H}$ is a set equipped with the binary operation $\circ_s$,*
   - *$\forall S, S' \in \mathbb{G}$: $f(S * S') = f(S) \circ_s f(S')$.*

   and an $I \in \mathbb{H}$, such that $f(s) = I$.

and `retrieve` *is an algorithm that inputs pk, m and the correctly converted pair (s, r) and retrieves*[5] *the signature σ on m.*

The class $\mathbb{S}$ differs from the class $\mathbb{C}$, introduced in [66], in the condition required for the function f. In fact, in our description of $\mathbb{S}$, the function f should satisfy a homomorphic property, whereas in the class $\mathbb{C}$, f should only possess an efficient protocol for proving knowledge of a preimage of a value in its range. We show in Theorem 9 that signatures in $\mathbb{S}$ accept also efficient proofs for proving knowledge of preimages, and thus belong to the class $\mathbb{C}$. Conversely, one can claim that signatures in $\mathbb{C}$ are also in $\mathbb{S}$, at least from a practical point of view, since it is not known in general how to achieve efficient protocols for proving knowledge of preimages of f without having the latter item satisfy some homomorphic properties. It is worth noting that similar to the classes $\mathbb{S}$ and $\mathbb{C}$ is the class of signatures introduced in [43], where the condition of having an efficient protocol for proving knowledge of preimages is weakened to having only a *witness hiding* proof of knowledge. Again, although this is a weaker assumption on f, all illustrations of signatures in this wider class happen to be also in $\mathbb{C}$ and $\mathbb{S}$. Our resort to specify the homomorphic property on f will be justified later when describing the confirmation/denial protocols of the resulting construction. In fact, these protocols are concurrent composition of proofs of knowledge and therefore need a careful study as it is known that zero knowledge is not closed under concurrent composition. Besides, the class $\mathbb{S}$ encompasses most proposals that were suggested so far, e.g. [4, 64, 37, 9, 59, 22, 13, 14, 6, 72, 70]. The reason why $\mathbb{S}$ includes most digital signature schemes lies in the fact that a signature verification consists in applying a function f to the "vital" part of the signature in question, then comparing the result to an expression computed from the message underlying the signature, the "auxiliary" or "simulatable" part of the signature, and finally the public parameters of the signature scheme. The function f must be one-way, otherwise the signature scheme is trivially forgeable. Moreover, it (f) consists most of the time of an arithmetic operation (e.g. exponentiation, raising to a power, pairing computation) which satisfies an easy homomorphic property.

Theorem 9 *The protocol depicted in Figure 3 is a Σ protocol for proving knowledge of preimages of the function f described in Definition 4.*

The proof is straightforward using the standard techniques. ∎

[5] Note that the `retrieve` algorithm suffices to ensure the non-triviality of the map f; given a pair (s, r) satisfying the conditions described in the definition, one can efficiently recover the original signature on the message.

Prover P		Verifier V
Choose $s' \xleftarrow{R} \mathbb{G}$		
Compute $t_2 = \Gamma.\mathtt{encrypt}(s')$	$\xrightarrow{\quad t_2 \quad}$	
	$\xleftarrow{\quad b \quad}$	Choose $b \xleftarrow{R} \{0,1\}^{\ell}$ $(b \in \mathbb{N})$
	$\xrightarrow{\quad z = s' * s^b \quad}$	
	$\xleftrightarrow{\mathsf{PoK}\{z = \Gamma.\mathtt{decrypt}_{\Gamma.sk}(t_2 \circ_e e^b)\}}$	
		Accept if the proof PoK is valid

Figure 4: Proof system for membership to the language $\{e\colon \exists m : m = \Gamma.\mathtt{decrypt}_{\Gamma.sk}(e)\}$ Common input: $(e, \Gamma.pk)$ and Private input: $\Gamma.sk$ or randomness encrypting m in e

6.3 The class $\mathbb{E}$ of encryption schemes

Definition 5 *$\mathbb{E}$ 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 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, c, and pk, there exists an efficient zero knowledge proof* PoK *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. $\forall m, m' \in \mathcal{M}$, $\forall pk\colon \Gamma.\mathtt{encrypt}_{pk}(m * m') = \Gamma.\mathtt{encrypt}_{pk}(m) \circ_e \Gamma.\mathtt{encrypt}_{pk}(m')$. *Moreover, given the randomness used to encrypt m in $\Gamma.\mathtt{encrypt}_{pk}(m)$ and m' in $\Gamma.\mathtt{encrypt}_{pk}(m')$, one can deduce (using only the public parameters) the randomness used to encrypt $m * m'$ in $\Gamma.\mathtt{encrypt}_{pk}(m) \circ_e \Gamma.\mathtt{encrypt}_{pk}(m')$.*

Examples of encryption schemes include for instance ElGamal's encryption [34], Paillier's encryption [55], or the Boneh-Boyen-Shacham scheme [7]. In fact, these schemes are homomorphic and possess an efficient proof of correctness of a decryption, namely the proof of equality of two discrete logarithms in case of [34, 7] and the proof of knowledge of an N-th root in case of [55].

Theorem 10 *Let Γ be an encryption scheme from the above class . Let furthermore c be an encryption of some message under some public key pk. The protocol depicted in Figure 4 is a Σ protocol for proving knowledge of the decryption of c if* PoK *consists of one move (i.e. is a non-interactive proof).*

The proof is again straightforward using the standard techniques. Note that the requirement on PoK is needed only to ensure that the overall proof runs in three moves (to qualify for a Σ protocol). We refer to Subsection 6.7 for techniques to efficiently turn a wide class of interactive proofs into non-interactive ones.

6.4 The class $\mathbb{C}$ of commitments

Definition 6 *$\mathbb{C}$ is the set of all commitment schemes for which there exists an algorithm* `compute` *that inputs the commitment public key pk, the message m and the commitment c on m, and computes a description of an injective*[6] *map $f : (\mathbb{G}, *) \to (\mathbb{H}, \circ_c)$ where:*

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

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 shown in Figure 5.

Theorem 11 *The protocol depicted in Figure 5 is a Σ protocol.* ∎

[6] Injectivity is needed to ensure the SpS property of the protocol depicted in Figure 5

Prover P		Verifier V
Compute I as defined in Definition 6		Compute I as defined in Definition 6
Choose $r' \xleftarrow{R} \mathbb{G}$		
Compute $t_1 = f(r')$	$\xrightarrow{\quad t_1 \quad}$	
	$\xleftarrow{\quad b \quad}$	Choose $b \xleftarrow{R} \{0,1\}^{\ell}$ $(b \in \mathbb{N})$
	$\xrightarrow{\quad z = r' * r^b \quad}$	
		Verify that $f(z) = t_1 \circ_c I^b$

Figure 5: Proof system for membership to the language $\{r\colon c = \mathtt{commit}(m, r)\}$ Common input: (c, m) and Private input : r.

Prover P		Verifier V
Compute I as defined in Definition 4		Compute I as defined in Definition 4
Choose $s' \xleftarrow{R} \mathbb{G}$		
Compute $t_1 = f(s')$		
Compute $t_2 = \Gamma.\mathtt{encrypt}_{\Gamma.pk}(s')$	$\xrightarrow{\quad t_1, t_2 \quad}$	
	$\xleftarrow{\quad b \quad}$	Choose $b \xleftarrow{R} \{0,1\}^{\ell}$ $(b \in \mathbb{N})$
	$\xrightarrow{\quad z = s' * s^b \quad}$	
	$\xleftrightarrow{\mathsf{PoK}\{z = \Gamma.\mathtt{decrypt}_{\Gamma.sk}(t_2 \circ_e (c,e)^b)\}}$	
		Accept if the proof PoK is valid and, $f(z) = t_1 \circ_s I^b$ in case of confirmation, $f(z) \neq t_1 \circ_s I^b$ in case of denial.

Figure 6: Proof system for membership to the language $\{(c, e, r)\colon \exists s : s = \Gamma.\mathtt{decrypt}(c, e) \wedge \Sigma.\mathtt{verify}(\mathtt{retrieve}(s, r), m \| c) = (\neq) 1\}$ Common input: $(c, e, r, \Sigma.pk, \Gamma.pk)$ and Private input: $\Gamma.sk$ or randomness encrypting s in (c, e)

6.5 Practical realizations from StE

We combine a secure signature scheme $\Sigma \in \mathbb{S}$ and a secure encryption scheme $\Gamma \in \mathbb{E}$, which is *derived from the KEM/DEM paradigm*, in the way described in Subsection 5.2. Namely we first compute an encapsulation c together with its corresponding key k. Then we compute a signature σ on c concatenated with the message to be signed. Finally convert σ to (s, r) using the `convert` algorithm described in Definition 4 and encrypt s in $e = \mathcal{D}.\mathtt{encrypt}_k(s)$ using k. The resulting confirmer signature is (c, e, r). We describe in Figure 6 the confirmation/denial protocols corresponding to the resulting construction. Note that the confirmation protocol can be also run by the signer who wishes to confirm the validity of a just generated confirmer signature using the randomness used to generate it.

Theorem 12 *The confirmation protocol (run by either the signer on a just generated signature or by the confirmer on any signature) described in Figure 6 is a Σ protocol if* PoK *is non-interactive.*

Proof The confirmation protocol depicted in Figure 6 is a parallel composition of the proofs depicted in Figures 3 and 4. Therefore completeness and soundness (SpS) follow as a direct consequence from the completeness and soundness of the underlying proofs (see [39]). Note that the SpS property rests on the *retrieve* property of Σ. In fact, the knowledge extractor will be able to extract a preimage of I, that together with r can be converted to a valid digital signature on m. Finally, the ZK simulator is the parallel composition of the ZK simulators for the mentioned protocols. ∎

Theorem 13 *The denial protocol described in Figure 6 is a Σ protocol with computational zero knowledge if Γ is IND-CPA-secure and* PoK *is non-interactive.*

Proof With the standard techniques, we prove that the denial protocol depicted in Figure 6 is complete and sound (with the SpS property, since the knowledge extractor will extract the (unique) decryption of (c, e) and check whether it is a valid digital signature on m or not). Similarly, we provide the following simulator to prove the ZK property.

1. Generate $b' \in_R \{0,1\}$. Choose $z \in_R \mathbb{G}$ and a random $t_1 \in_R f(\mathbb{G})$ and $t_2 = \Gamma.\mathtt{encrypt}_{\Gamma.pk}(z) \circ_e (c,e)^{-b}$.
2. Get b from the verifier. If $b = b'$, it sends z and simulates the proof PoK of z being the decryption of $t_2 \circ_e (e, s_k)^b$. If $b \neq b'$, it goes to Step 1.

The prover's first message is an encryption of a random value $s' \in_R \mathbb{G}$, in addition to $f(s')$. The simulator's first message is an encryption of a random value $z * s$ and the element $t_1 \in_R f(\mathbb{G})$ (independent of z). Distinguishing these two cases is at least as hard as breaking the IND-CPA security of the underlying encryption scheme. In fact, if the verifier is able to distinguish these two cases, it can be easily used to break the encryption scheme in the IND-CPA sense. Therefore, under the assumption of the IND-CPA security of the encryption scheme, the simulator's and prover's first message distributions are indistinguishable. Moreover, the simulator runs in expected polynomial time, since the number of rewinds is 2^ℓ. Finally, the distributions of the prover's and the simulator's messages in the last round are again, by the same argument, indistinguishable under the IND-CPA security of the encryption scheme. ∎

Remark 11 *The above confirmation/denial protocols can be efficiently turned into Σ protocols with concurrent zero knowledge. We refer to [25] for the details of this transformation. Moreover, these protocols can also be efficiently turned into designated verifier proofs of knowledge using the techniques described in Subsection 6.1.*

Before ending this subsection, we remark that our new StE captures many efficient realizations of confirmer/undeniable signatures proposed recently, e.g. [50, 65]. It also serves for analyzing some early schemes that had a speculative security: the Damgård-Pedersen [24] undeniable signatures. In fact, these signatures had a security analysis that was left open for over than a decade. We defer in Appendix A the analysis of this scheme. More precisely, We disprove the claimed invisibility of these signatures in the model given in [24]. Next, we propose a fix to these signatures so that they become invisible; interestingly, this repair turns out to be a special instantiation of our new StE paradigm. Actually, even the confirmation/denial protocols provided in [24] happen to be a special case of the confirmation/denial protocols depicted in Figure 6.

6.6 Practical realizations from CtEaS

The (new) CtEaS has the merit of supporting any digital signature scheme as a building block. In fact, efficient as the StE paradigm is, it still applies only to a restricted class of signatures. For instance, StE does not seem to be plausible with the PSS signature scheme [4].

CtEaS does not involve a proof of knowledge of a signature in its confirmation/denial protocols. In fact, confirmation (denial) of a signature on a certain message consists in proving knowledge of the decryption of a given ciphertext, and that this decryption is (is not) the opening value of a given commitment on the message in question. More specifically, the confirmation/denial protocols for CtEaS, when the encryption scheme Γ belongs to the class $\mathbb{E}$ and the commitment scheme Ω belongs to the class $\mathbb{C}$, are depicted in Figure 7.

Theorem 14 *The confirmation protocol described in Figure 7 is a Σ protocol if* PoK *is non-interactive.*

Theorem 15 *The denial protocol described in Figure 7 is a Σ protocol with computational zero knowledge if Γ is IND-CPA-secure and* PoK *is non-interactive.*

The proofs of both Theorem 14 and Theorem 15 are similar to those of Theorem 12 and Theorem 13 respectively. ∎

6.7 The EtS paradigm

EtS can be seen as a special instance of EtCaS as IND-CPA encryption can be easily used to get statistically binding and computationally hiding commitments. Therefore, one can first commit to the message to be signed using the the encryption scheme, then sign the resulting ciphertext. The confirmer signature is composed of the ciphertext and of its encryption. In fact, there will be no need to encrypt the string used to produce the ciphertext, since the private key of the encryption scheme is sufficient to check the validity of a ciphertext w.r.t. a given message. Finally, selective conversion is achieved by releasing a ZK *non-interactive* proof that the ciphertext (first part of the confirmer signature) decrypts to the message in question.

Prover P		Verifier V
Compute I as defined in Definition 6		Compute I as defined in Definition 6
Choose $r' \xleftarrow{R} \mathbb{G}$		
Compute $t_1 = f(r')$		
Compute $t_2 = \Gamma.\mathtt{encrypt}_{\Gamma.pk}(r')$	$\xrightarrow{t_1, t_2}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^\ell$ $(b \in \mathbb{N})$
	$\xrightarrow{z = r' * r^b}$	
	$\xleftrightarrow{\mathsf{PoK}\{z = \Gamma.\mathtt{decrypt}_{\Gamma.sk}(t_2 \circ_e e^b)\}}$	
		Accept if PoK is valid and, $f(z) = t_1 \circ_c I^b$ in case of confirmation, $f(z) \neq t_1 \circ_c I^b$ in case of denial.

Figure 7: Proof system for membership to the language $\{(e,c)\colon \exists r : r = \Gamma.\mathtt{decrypt}(e) \wedge c = (\neq)\Omega.\mathtt{commit}(m,r)\}$ Common input: $(e, c, m, \Gamma.pk, \Omega.pk)$ and Private input: $\Gamma.sk$ or randomness encrypting r in e.

Prover P		Verifier V
Choose $m' \xleftarrow{R}$		
Compute $t_1 = f(m')$		
Compute $t_2 = \Gamma.\mathtt{encrypt}_{\Gamma.pk}(m')$	$\xrightarrow{t_1, t_2}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^\ell$ $(b \in \mathbb{N})$
	$\xrightarrow{z = m' * \tilde{m}^b}$	
	$\xleftrightarrow{\mathsf{PoK}\{z = \Gamma.\mathtt{decrypt}_{\Gamma.sk}(t_2 \circ c^b)\}}$	
		Accept if PoK is valid and $f(z) \neq t_1 \circ f(m)^b$.

Figure 8: Proof system for membership to the language $\{(m,c)\colon \exists \tilde{m} : \tilde{m} = \Gamma.\mathtt{decrypt}(c) \wedge \tilde{m} \neq m\}$ Common input: $(m, c, \Gamma.pk)$ and Private input: $\Gamma.sk$ or randomness encrypting $\tilde{m}$ in c

Similarly to CtEaS, unforgeability of the construction rests on the unforgeability of the underlying signature scheme, whereas invisibility rests on the strong unforgeability (SEUF-CMA) of the signature and on the indistinguishability (IND-CPA) of the underlying encryption. We discuss in the rest of this subsection how to achieve practical realizations from this technique.

6.7.1 Confirmation/denial protocols

Confirmation in EtS amounts to a proof of correctness of a decryption (i.e. that a given ciphertext encrypts a given message). This is in general easy since in most encryption schemes, one can define, given a ciphertext c and its underlying plaintext m, two homomorphic maps f and g, and two quantities I and J such that $f(r) = I$ and $g(sk) = J$, where r is the randomness used to encrypt m in c, and sk is the private key of the encryption scheme in question. Examples of such encryptions include [34, 7, 55, 23, 16]. The confirmation (`sconfirm`) protocol in this case will be reduced to a proof of knowledge of a preimage of J (I) by the function g (f), for which we provided an efficient proof in Figure 3.

Concerning the denial protocol, it is not always straightforward. In most discrete-logarithm-based encryptions, this protocol amounts to a proof of inequality of discrete logarithms as in [34, 7, 23]. In case the encryption scheme belongs to the class $\mathbb{E}$ defined earlier, Figure 8 provides an efficient proof that c encrypts some $\tilde{m} \neq m$. In the protocol provided in this figure, f denotes an arbitrary injective map:

$$f(m * m') = f(m) \circ f(m')$$

With the standard tools, the above denial protocol can be shown to be a Σ protocol with computational ZK, if the encryption scheme Γ is IND-CPA secure, and ZK closed under concurrent composition if PoK is non-interactive.

6.7.2 Selective conversion

Selective conversion in confirmer signatures from EtS consists of a non-interactive proof of the confirmation protocol. We note in this paragraph three solutions to achieve this goal:

The case of Paillier [55]'s encryption scheme. This scheme operates on messages in $\mathbb{Z}_N$, where $N = pq$ is a safe RSA modulus. Encryption of a message m is done by picking a random $r \in_R \mathbb{Z}_N^\times$ and then computing the ciphertext $c = r^N(1 + mN) \bmod N^2$. Decryption of a ciphertext c is done by raising it to $\lambda = \mathrm{lcm}(p-1, q-1)$ to find m. It is easy to see that Paillier's encryption belongs to what we call the class of *fully decryptable* encryption schemes, i.e. encryption schemes where decryption leads to the randomness used to produce the ciphertext. Thus, selective conversion can simply be achieved by releasing the randomness used to generate the ciphertext.

Damgård et al. [26]'s solution. This solution transforms a 3-move interactive ZK protocol P with linear answer to a non-interactive ZK one (NIZK) using a homomorphic encryption scheme in a registered key model, i.e. in a model where the verifier registers his key. The authors in [26] proposed an efficient illustration using Paillier's encryption and the proof of equality of two discrete logarithms. We conclude that with such a technique, EtS accepts an efficient instantiation if the considered encryption allows proving the correctness of a decryption using the proof of equality of two discrete logarithms, e.g. [34, 7, 23].

Groth and Sahai [44]'s solution. This technique is applicable in general for encryption schemes where the encryption/decryption algorithms perform only group or pairing (if bilinear groups are involved) operations on the randomness or the private key resp. In fact, [44] provides efficient NIZK for languages that cover the aforementioned operations.

7 Verifiable Signcryption

7.1 Syntax and model

A verifiable signcryption scheme consists of the following algorithms/protocols:

Setup ($\mathtt{setup}(1^\kappa)$). This probabilistic algorithm inputs a security parameter κ, and generates the public parameters *param* of the signcryption scheme.

Key generation ($\mathtt{keygen}_U(1^\kappa, param), U \in \{S, R\}$). This probabilistic algorithm inputs the security parameter κ and the public parameters *param*, and outputs a key pair (pk_U, sk_U) for the system user U which is either the sender S or the receiver R.

Signcryption ($\mathtt{signcrypt}(m, sk_S, pk_S, pk_R)$). This probabilistic algorithm inputs a message m, the key pair (sk_S, pk_S) of the sender, the public key pk_R of the receiver, and outputs the signcryption μ of the message m.

Proof of validity ($\mathtt{proveValidity}(\mu, pk_S, pk_R)$). This is an interactive protocol between the receiver or the sender who has just generated a signcryption μ on some message, and any verifier: the sender uses the randomness used to create μ (as private input) and the receiver uses his private key sk_R in order to convince the verifier that μ is a valid signcryption on some message. The common input to both the prover and the verifier comprise the signcryption μ in question, pk_S, and pk_R. At the end of the protocol, the verifier either accepts or rejects the proof.

Unsigncryption ($\mathtt{unsigncrypt}(\mu, sk_R, pk_R, pk_S)$). This is a deterministic algorithm which inputs a putative signcryption μ on some message, the key pair (sk_R, pk_R) of the receiver, and the public key pk_S of the sender, and outputs either the message underlying μ or an error symbol $\perp$.

Confirmation/Denial ($\{\mathtt{confirm}, \mathtt{deny}\}(\mu, m, pk_R, pk_S)$). These are interactive protocols between the receiver and any verifier; the receiver uses his private key sk_R (as private input) to convince any verifier that a signcryption μ on some message m is/is not valid. The common input comprises the signcryption μ and the message m in question, in addition to pk_R and

Experiment $\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{A}}(1^\kappa)$

1. $param \leftarrow sc.\mathtt{setup}(1^\kappa)$;
2. $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow sc.\mathtt{keygen}_S(1^\kappa, param)$;
3. $(pk_\mathsf{R}, sk_\mathsf{R}) \leftarrow \mathcal{A}(pk_\mathsf{S})$;
4. $\mu^\star \leftarrow \mathcal{A}^{\mathfrak{S}}(pk_\mathsf{S}, pk_\mathsf{R}, sk_\mathsf{R})$;
   $\mathfrak{S} : m \longmapsto sc.\mathtt{signcrypt}\{sk_\mathsf{S}, pk_\mathsf{S}, pk_\mathsf{R}\}(m)$
5. return 1 if and only if :
   - $sc.\mathtt{unsigncrypt}_{\{sk_\mathsf{R}, pk_\mathsf{R}, pk_\mathsf{S}\}}[\mu^\star] = m^\star$
   - $m^\star$ was not queried to $\mathfrak{S}$

pk_S. At the end of the protocol, the verifier is either convinced of the validity/invalidity of μ w.r.t. m or not.

Public verification ($\mathtt{publicVerify}(\mu, m, sk_R, pk_R, pk_S)$). This is an algorithm which inputs a signcryption μ, a message m, the key pair (sk_R, pk_R) of the receiver, and the public key pk_S of the sender, and outputs either an error symbol $\perp$ if μ is not a valid signcryption on m, or a string which allows to publicly verify the validity of μ on m otherwise.

It is natural to require the correctness of a signcryption scheme, i.e. for any message m:

$$\mathtt{unsigncrypt}(\mathtt{signcrypt}(m, sk_S, pk_S, pk_R), sk_R, pk_R, pk_S) = m.$$

and

$$\mathtt{publicVerify}(m, \mathtt{signcrypt}(m, sk_S, pk_S, pk_R), sk_R, pk_R, pk_S) \neq \perp .$$

Moreover, the protocols proveValidity and {confirm, deny} must be complete, sound, and non-transferable.

Finally, we require in a signcryption scheme two further properties, namely unforgeability and indistinguishability which we detail in the rest of this section.

7.1.1 Unforgeability

This notion protects the sender's authenticity from *malicious insider* adversaries, i.e. the receiver. It is defined through a game between a challenger C and an adversary $\mathcal{A}$ where the latter gets the public key pk_S of the sender, generates the key pair $(pk_\mathsf{R}, sk_\mathsf{R})$ of the receiver, and hands pk_R to the challenger. During the game, $\mathcal{A}$ is allowed to ask adaptively signcryption queries w.r.t. pk_R and pk_S on messages of his choice to C. The scheme is said to be *Existentially Unforgeable against Chosen Message Attacks (EUF-CMA)* if the adversary is unable to produce a valid signcryption $\mu^\star$ on a message $m^\star$ that he did not ask to the signcryption oracle.

Definition 7 (Unforgeability) *We consider a signcryption scheme sc given by the algorithms/protocols defined earlier in this section. Let $\mathcal{A}$ be a PPTM. We consider the following random experiment:*

We define the success *of $\mathcal{A}$* via*:*

$$\mathbf{Succ}^{\textit{euf-cma}}_{\mathcal{A}}(1^\kappa) = \Pr\left[\mathbf{Exp}^{\textit{euf-cma}}_{\mathcal{A}}(1^\kappa) = 1\right].$$

Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, $\mathcal{A}$ is called a (t, ε, q_s)-EUF-CMA adversary against if, running in time t and issuing q_s queries to the .signcrypt *oracle, $\mathcal{A}$ has $\mathbf{Succ}^{\textit{euf-cma}}_{\mathcal{A}}(1^\kappa) \geq \varepsilon$. The scheme is said to be (t, ε, q_s)-EUF-CMA secure if no (t, ε, q_s)-EUF-CMA adversary against it exists.*

Remark 12 *Note that $\mathcal{A}$ is not given the* proveValidity, unsigncrypt, publicVerify, *and* {confirm, deny} *oracles. In fact, these oracles are useless for him as he has the receiver's private key sk_R at his disposal.*

Experiment $\mathbf{Exp}^{\text{ind-cca-}b}_{\mathcal{A}}(1^\kappa)$

1. $param \leftarrow sc.\mathtt{setup}(1^\kappa)$;
2. $(sk_{\mathsf{S}}, pk_{\mathsf{S}}) \leftarrow sc.\mathtt{keygen}_{S}(1^\kappa, param)$;
3. $(sk_{\mathsf{R}}, pk_{\mathsf{R}}) \leftarrow sc.\mathtt{keygen}(1^\kappa, param)$;
4. $(m_0^\star, m_1^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{V},\mathfrak{U},\mathfrak{C}}(\mathsf{find}, pk_{\mathsf{S}}, pk_{\mathsf{R}})$;
$$\left|\begin{array}{l}\mathfrak{S} : m \longmapsto sc.\mathtt{signcrypt}_{\{sk_{\mathsf{S}},pk_{\mathsf{S}},pk_{\mathsf{R}}\}}(m)\\ \mathfrak{V} : \mu \longmapsto sc.\mathtt{proveValidity}(\mu, pk_{\mathsf{S}}, pk_{\mathsf{R}})\\ \mathfrak{U} : \mu \longmapsto sc.\mathtt{unsigncrypt}_{sk_{\mathsf{R}},pk_{\mathsf{R}},pk_{\mathsf{S}}}(\mu)\\ \mathfrak{C} : (\mu, m) \longmapsto sc.\{\mathtt{confirm}, \mathtt{deny}\}(\mu, m, pk_{\mathsf{R}}, pk_{\mathsf{S}})\\ \mathfrak{P} : (\mu, m) \longmapsto sc.\mathtt{publicVerify}(\mu, m, pk_{\mathsf{R}}, pk_{\mathsf{S}})\end{array}\right.$$
5. $\mu^\star \leftarrow sc.\mathtt{signcrypt}_{\{sk_{\mathsf{S}},pk_{\mathsf{S}},pk_{\mathsf{R}}\}}(m_b^\star)$;
6. $d \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{V},\mathfrak{U},\mathfrak{C}}(\mathsf{guess}, \mathcal{I}, \mu^\star, pk_{\mathsf{S}}, pk_{\mathsf{C}})$;
$$\left|\begin{array}{l}\mathfrak{S} : m \longmapsto sc.\mathtt{signcrypt}_{\{sk_{\mathsf{S}},pk_{\mathsf{S}},pk_{\mathsf{R}}\}}(m)\\ \mathfrak{V} : \mu \longmapsto sc.\mathtt{proveValidity}(\mu, pk_{\mathsf{S}}, pk_{\mathsf{R}})\\ \mathfrak{U} : \mu(\neq \mu^\star) \longmapsto sc.\mathtt{unsigncrypt}_{sk_{\mathsf{R}},pk_{\mathsf{R}},pk_{\mathsf{S}}}(\mu)\\ \mathfrak{C} : (\mu, m)(\neq (\mu^\star, m_i^\star), i = 0, 1) \longmapsto sc.\{\mathtt{confirm}, \mathtt{deny}\}(\mu, m, pk_{\mathsf{R}}, pk_{\mathsf{S}})\\ \mathfrak{P} : (\mu, m)(\neq (\mu^\star, m_i^\star), i = 0, 1) \longmapsto sc.\mathtt{publicVerify}(\mu, m, pk_{\mathsf{R}}, pk_{\mathsf{S}})\end{array}\right.$$
7. Return d;

7.1.2 Indistinguishability

This notion protects the sender's privacy from *outsider adversaries*. It is defined through a game between a challenger C and an adversary $\mathcal{A}$; C generates the key pairs $(sk_{\mathsf{S}}, pk_{\mathsf{S}})$ and $(sk_{\mathsf{R}}, pk_{\mathsf{R}})$ for the sender and for the receiver respectively, and hands $(pk_{\mathsf{S}}, pk_{\mathsf{R}})$ to $\mathcal{A}$. During the first phase of the game, $\mathcal{A}$ queries adaptively `signcrypt` and `proveValidity` (actually `proveValidity` is only invoked on inputs just obtained from the signcryption oracle), `unsigncrypt`, {`confirm`, `deny`}, and `publicVerify` for any input. Once $\mathcal{A}$ decides that this phase is over, he generates two messages $m_0^\star, m_1^\star$ and hands them to C who generates a signcryption $\mu^\star$ on $m_b^\star$ for $b \xleftarrow{R} \{0, 1\}$ and gives it $(\mu^\star)$ to $\mathcal{A}$. The latter resumes querying the previous oracles adaptively on any input with the exception of not querying `unsigncrypt` on $\mu^\star$, and {`confirm`, `deny`} and `publicVerify` on the pair $(\mu^\star, m_i^\star)$ for $i \in \{0, 1\}$. At the end, the adversary outputs his guess b' for the message underlying the signcryption $\mu^\star$. He is considered successful if $b = b'$.

Definition 8 (Indistinguishability (IND-CCA)) *Let sc be a signcryption scheme, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment for $b \xleftarrow{R} \{0, 1\}$:*
We define the advantage *of $\mathcal{A}$* via*:*

$$\mathbf{Adv}^{\textit{ind-cca}}_{\mathcal{A}}(1^\kappa) = \left|\Pr\left[\mathbf{Exp}^{\text{ind-cca-b}}_{\mathcal{A}}(1^\kappa) = b\right] - \frac{1}{2}\right|.$$

Given $(t, q_s, q_v, q_u, q_{cd}, q_{pv}) \in \mathbb{N}^6$ and $\varepsilon \in [0, 1]$, $\mathcal{A}$ is called a $(t, \varepsilon, q_s, q_v, q_u, q_{cd}, q_{pv})$-IND-CCA adversary against if, running in time t and issuing q_s queries to the `signcrypt` *oracle, q_v queries to the* `proveValidity` *oracle, q_u queries to the* `unsigncrypt` *oracle, q_{cd} queries to the* {`confirm`, `deny`} *oracle, and q_{pv} to the* `publicVerify` *oracle, $\mathcal{A}$ has $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{A}}(1^\kappa) \geq \varepsilon$. The scheme is said to be $(t, \varepsilon, q_s, q_v, q_u, q_{cd}, q_{pv})$-IND-CCA secure if no $(t, \varepsilon, q_s, q_v, q_u, q_{cd}, q_{pv})$-IND-CCA adversary against it exists.*

7.2 Classical constructions for verifiable signcryption

Let Σ be a digital signature scheme given by Σ.`keygen` which generates a key pair ($\Sigma.sk$, $\Sigma.pk$), Σ.`sign`, and Σ.`proveValidity`. Let furthermore Γ denote a public key encryption scheme described by Γ.`keygen` that generates the key pair ($\Gamma.sk$,$\Gamma.pk$), Γ.`encrypt`, and Γ.`decrypt`. Finally, let Ω be a commitment scheme given by the algorithms Ω.`commit` and Ω.`open`. The most popular paradigms used to devise signcryption schemes from basic primitives are:

- The *"sign-then-encrypt" (StE) paradigm*. Given a message m, `signcrypt` first produces a signature σ on the message using $\Sigma.sk$, then encrypts $m\|\sigma$ under $\Gamma.pk$. The result forms the signcryption on m. To `unsigncrypt`, one first decrypts the signcryption using $\Gamma.sk$ in $m\|\sigma$, then checks the validity of σ, using $\Sigma.pk$, on m. Finally, `publicVerify` of a valid signcryption $\mu = \Gamma.\texttt{encrypt}(m\|\sigma)$ on m outputs σ.

- The *"encrypt-then-sign" (EtS) paradigm*. Given a message m, `signcrypt` produces an encryption e on m using $\Gamma.pk$, then produces a signature σ on e using $\Sigma.sk$; the signcryption is the pair (e, σ). To `unsigncrypt` such a signcryption, one first checks the validity of σ w.r.t. e using $\Sigma.pk$, then decrypts e using $\Gamma.sk$ to get m. Finally, `publicVerify` outputs a zero knowledge non-interactive (NIZK) proof that m is the decryption of e; such a proof is possible since the statement in question is in NP ([40] and [5]).

- The *"commit-then-encrypt-and-sign" (CtEaS) paradigm*. This construction has the advantage of performing the signature and the encryption *in parallel* in contrast to the previous sequential compositions. Given a message m, one first produces a commitment c on it using some random nonce r, then encrypts $m\|r$ under $\Gamma.pk$, *and* produces a signature σ on c using $\Sigma.sk$. The signcryption is the triple (e, c, σ). To `unsigncrypt` such a signcryption, one first checks the validity of σ w.r.t. c, then decrypts e to get $m\|r$, and finally checks the validity of the commitment c w.r.t (m, r). `publicVerify` is achieved by releasing the decryption of e, namely $m\|r$.

The proofs of well (mal) formed-ness, namely `proveValidity` and {`confirm`, `deny`} can be carried out since the languages in question are in NP and thus accept zero knowledge proof systems [40]. Finally, it is possible to require a proof in the `publicVerify` algorithms of StE and CtEaS, that the revealed information is indeed a correct decryption of the encryption in question; such a proof is again possible to issue since the corresponding statement is in NP.

7.3 Negative results for StE and CtEaS

We proceed in this subsection as we did in confirmer signatures. We first introduce a weaker notion of indistinguishability that we denote by WIND-CCA. Then we prove that OW-CCA and NM-CPA secure encryption are insufficient to yield StE and CtEaS constructions meeting this notion, and thus all other stronger notions of indistinguishability, for instance the traditional IND-CCA notion. We first prove this result for *key-preserving reductions*, then we generalize it to arbitrary reductions assuming further properties on the underlying encryption. Finally, we rule out the OW-CPA, IND-CPA, and the OW-PCA notion by remarking that ElGamal's [34] encryption scheme meets all those notions but leads to a simple attack against IND-CCA, when employed in constructions from StE and CtEaS.

Weak indistinguishability (WIND-CCA) The WIND-CCA notion is defined along the same lines of IND-CCA except at the challenge phase; in this notion the adversary outputs a message $m^\star$ and gets as a challenge a signcryption $\mu^\star$ of either $m^\star$ or a random message generated by the challenger. There is the natural restriction of not querying $\mu^\star$ to the `unsigncrypt` or `proveValidity` oracles, or $(m^\star, \mu^\star)$ to the {`confirm`, `deny`} or `publicVerify` oracles.

It is obvious that an adversary against WIND-CCA can be easily transformed into an adversary against the IND-CCA of the same signcryption scheme.

Lemma 5 *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts a WIND-CCA adversary $\mathcal{A}$ against signcryptions from the StE (CtEaS) paradigm to a OW-CCA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that OW-CCA breaks the encryption scheme in question.*

Proof Let $\mathcal{R}$ be the key-preserving reduction that reduces OW-CCA breaking the encryption scheme underlying the construction to WIND-CCA breaking the construction (from StE or CtEaS) itself. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to OW-CCA break the same encryption scheme by simulating an execution of the WIND-CCA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack. $\mathcal{M}$ gets his challenge key $\Gamma.pk$ and challenge ciphertext c. He is further equipped with a decryption oracle that he can query on all ciphertexts of

$\boxed{\text{Experiment } \mathbf{Exp}^{\mathsf{WIND\text{-}CCA}}_{\mathcal{A}}(1^\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(sk_{\mathsf{S}}, pk_{\mathsf{S}}) \leftarrow \mathtt{keygen}_((1^\kappa, param)$;
3. $(sk_{\mathsf{R}}, pk_{\mathsf{R}}) \leftarrow \mathtt{keygen}(1^\kappa, param)$;
4. $(m_0^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{V},\mathfrak{U},\mathfrak{C}}(\mathsf{find}, pk_{\mathsf{S}}, pk_{\mathsf{R}})$;

   $\mathfrak{S} : m \longmapsto \mathtt{signcrypt}_{\{sk_{\mathsf{S}}, pk_{\mathsf{S}}, pk_{\mathsf{R}}\}}(m)$
   $\mathfrak{V} : \mu \longmapsto \mathtt{proveValidity}(\mu, pk_{\mathsf{S}}, pk_{\mathsf{R}})$
   $\mathfrak{U} : \mu \longmapsto \mathtt{unsigncrypt}_{sk_{\mathsf{R}}, pk_{\mathsf{R}}, pk_{\mathsf{S}}}(\mu)$
   $\mathfrak{C} : (\mu, m) \longmapsto \{\mathtt{confirm}, \mathtt{deny}\}(\mu, m, pk_{\mathsf{R}}, pk_{\mathsf{S}})$
   $\mathfrak{P} : (\mu, m) \longmapsto \mathtt{publicVerify}(\mu, m, sk_{\mathsf{R}}, pk_{\mathsf{R}}, pk_{\mathsf{S}})$
5. $m_1^\star \xleftarrow{R} \mathsf{M}$; $b \xleftarrow{R} \{0, 1\}$; $\mu^\star \leftarrow \mathtt{signcrypt}\{sk_{\mathsf{S}}, pk_{\mathsf{S}}, pk_{\mathsf{R}}\}(m_b^\star)$
6. $b^\star \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{V},\mathfrak{U},\mathfrak{C}}(\mathsf{guess}, \mathcal{I}, \mu^\star, pk_{\mathsf{S}}, pk_{\mathsf{R}})$

   $\mathfrak{S} : m \longmapsto \mathtt{signcrypt}_{\{sk_{\mathsf{S}}, pk_{\mathsf{S}}, pk_{\mathsf{R}}\}}(m)$
   $\mathfrak{V} : \mu \longmapsto \mathtt{proveValidity}(\mu, pk_{\mathsf{S}}, pk_{\mathsf{R}})$
   $\mathfrak{U} : \mu(\neq \mu^\star) \longmapsto \mathtt{unsigncrypt}_{sk_{\mathsf{R}}, pk_{\mathsf{R}}, pk_{\mathsf{S}}}(\mu)$
   $\mathfrak{C} : (\mu, m)(\neq (\mu^\star, m_0^\star)) \longmapsto \{\mathtt{confirm}, \mathtt{deny}\}(\mu, m, pk_{\mathsf{R}}, pk_{\mathsf{S}})$
   $\mathfrak{P} : (\mu, m)(\neq (\mu^\star, m_0^\star)) \longmapsto \mathtt{publicVerify}(\mu, m, sk_{\mathsf{R}}, pk_{\mathsf{R}}, pk_{\mathsf{S}})$
7. If $(b = b^\star)$ return 1 else return 0.

his choice except c. $\mathcal{M}$ launches $\mathcal{R}$ over Γ with the same public key $\Gamma.pk$ and the same challenge c. Obviously, all decryption queries made by $\mathcal{R}$, which are by definition different from the challenge c, can be forwarded to $\mathcal{M}$'s own challenger. $\mathcal{M}$ needs now to simulate a WIND-CCA adversary $\mathcal{A}$ to $\mathcal{R}$ (Σ and Ω denote respectively the signature and the commitment schemes in use):

- *StE paradigm.* $\mathcal{A}$ outputs a message $m^\star$ and receives a challenge $\mu^\star$ which is either a valid signcryption on $m^\star$ or a signcryption on a random message chosen by $\mathcal{R}$. Two cases manifest, either $\mu^\star$ is different from c, in which case $\mathcal{M}$ can perfectly solve his challenge by querying his own challenger for the decryption of $\mu^\star$. Or $\mu^\star$ is equal to c; in this case $\mathcal{M}$ will answer that $\mu^\star$ is not a valid signcryption on $m^\star$. In fact, with overwhelming probability, the OW-CCA challenge c is not a valid signcryption on a message chosen later by $\mathcal{M}$ (upon receipt of the challenge). Therefore, $\mathcal{M}$ (in both cases) can correctlty simulate the WIND-CCA attacker to $\mathcal{R}$.
- *CtEaS paradigm.* $\mathcal{A}$ outputs a message $m^\star$ and receives a challenge $\mu^\star = (e^\star, c^\star, \sigma^\star)$ which is a signcryption on either $m^\star$ or a random message chosen by $\mathcal{R}$. Similarly, with overwhelming probability c is not encryption, under $\Gamma.pk$, of a string whose prefix is $m^\star$. Therefore, $\mathcal{A}$ is able to answer perfectly his challenge; if $c = e^\star$, then $\mathcal{A}$ responds that $\mu^\star$ is not a valid signcryption of $m^\star$, otherwise $\mathcal{A}$ solves the challenge by querying $e^\star$ to own his decryption oracle for Γ.

To sum up, $\mathcal{M}$ is able to perfectly answer the decryption queries made by $\mathcal{R}$ (that are by definition different from c). $\mathcal{M}$ is further capable of successfully simulating the WIND-CCA adversary against the construction (from either StE or CtEaS). Thus $\mathcal{R}$ is expected to return the answer to the OW-CCA challenge, namely the decryption of c. Upon receipt of this answer, $\mathcal{M}$ will forward it to his own challenger. ∎

We further have this lemma on the insufficiency of NM-CPA secure encryption.

Lemma 6 *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an IND-CCA adversary $\mathcal{A}$ against signcryptions from the StE (CtEaS) paradigm to an NM-CPA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that NM-CPA breaks the encryption scheme in question.*

Proof Let $\mathcal{R}$ be the key-preserving reduction that reduces NM-CPA breaking the encryption scheme underlying the construction to IND-CCA breaking the construction (from the StE or EtS) itself. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to NM-CPA break the same encryption scheme by simulating an execution of the IND-CCA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack.

- *StE paradigm.* $\mathcal{M}$ (behaving as $\mathcal{A}$) will query $\mathcal{R}$ on two messages m_0, m_1 ($m_0 \neq m_1$) for signcryption. Let μ_0, μ_1 be the corresponding signcryptions respectively. $\mathcal{M}$ will query again (μ_i, m_i), $i \in \{0,1\}$, for public verification. Let σ_0, σ_1 be the corresponding answers respectively. At that point, $\mathcal{M}$ outputs $D = \{m_0\|\sigma_0, m_1\|\sigma_1\}$, to his NM-CPA challenger, as a distribution probability from which the messages will be drawn. He gets in response a challenge encryption $\mu^\star$, of either $m_0\|\sigma_0$ or $m_1\|\sigma_1$ under $\Gamma.pk$, and is asked to produce a ciphertext μ' whose corresponding plaintext is meaningfully related to the decryption of $\mu^\star$. To solve his task, $\mathcal{M}$ queries $\mu^\star$ for unsigncryption. Let m_b be the result of such a query with $b \in \{0,1\}$. $\mathcal{M}$ will output $\Gamma.\mathtt{encrypt}_{\Gamma.pk}(\overline{m_b\|\sigma_b})$ ($\overline{m}$ refers to the bit-complement of the element m) and the relation R: $R(m,m') = (m' = \overline{m})$. Finally $\mathcal{M}$ aborts the game (stops simulating an IND-CCA attacker against the generic construction).

- *CtEaS paradigm.* Similarly, $\mathcal{M}$ queries $\mathcal{R}$ on m_0, m_1 ($m_0 \neq m_1$) for signcryption. Let $\mu_0 = (e_0, c_0, \sigma_0)$ and $\mu_1 = (e_1, c_1, \sigma_1)$ be the corresponding signcryptions. $\mathcal{M}$ will then query μ_0, μ_1, along with the corresponding messages, for public verification. Let $m_0\|r_0$ and $m_1\|r_1$ be the corresponding answers. $\mathcal{M}$ will output $D = \{m_0\|r_0, m_1\|r_1\}$, to his NM-CPA challenger, as a distribution probability from which the messages will be drawn. He will receive a challenge encryption $e^\star$, of either $m_0\|r_0$ or $m_1\|r_1$. $\mathcal{M}$ will then query the unsigncryption of $(e^\star, c_b, \sigma_b)$ for some $b \xleftarrow{R} \{0,1\}$. If the answer is different from $\perp$, then $e^\star$ is indeed an encryption of $m_b\|r_b$, and thus $\mathcal{M}$ will output $\Gamma.\mathtt{encrypt}_{\Gamma.pk}(\overline{m_b\|r_b})$ and the relation R (defined above). Otherwise, $\mathcal{M}$ will output $\Gamma.\mathtt{encrypt}_{\Gamma.pk}(\overline{m_{1-b}\|r_{1-b}})$ and the same relation R. Finally $\mathcal{M}$ aborts the game (stops simulating an IND-CCA attacker against the generic construction).

We generalize the previous results to arbitrary reductions as in Subsection 4.3 if the encryption scheme has a *non-malleable key generator*, which informally means that OW-CCA (NM-CPA) breaking the encryption, w.r.t. a public key pk, is no easier when given access to a decryption oracle w.r.t. any key pk' different from pk.

Moreover, we can rule out the OW-CPA, OW-PCA, and IND-CPA notions by remarking that ElGamal's encryption meets all those notions (under different assumptions), but cannot be employed in StE and CtEaS as it is malleable. In fact, the indistinguishability adversary can create a new signcryption (by re-encrypting the ElGamal encryption) on the challenge message, and query it for unsigncryption. The answer of such a query is sufficient to conclude.

In consequence of the above analysis, the used encrypted scheme has to satisfy at least IND-PCA security in order to lead to secure signcryption from StE or CtEaS. Since there are no known encryption schemes in the literature which separate the notions IND-PCA and IND-CCA, our result practically means that the encryption scheme underlying the previous constructions has to satisfy the highest security level (IND-CCA) in order to lead to secure signcryption. This translates in expensive operations, especially if verifiability is further required for the resulting signcryption.

7.4 Positive results for StE and CtEaS

Signcryptions from StE or CtEaS suffer the strong forgeability: given a signcryption on some message, one can create another valid signcryption on the same message without the sender's help. To circumvent this problem, we apply the same techniques used previously in confirmer signatures, namely bind the digital signature to its corresponding signcryption. This translates for CtEaS in producing the digital signature on both the commitment and the encryption (of the message in question and the opening value of the commitment). Similarly to confirmer signatures, the new CtEaS looses the parallelism of the original one, i.e. encryption and signature can no longer be carried out in parallel, however it has the advantage of resting on cheap encryption compared to the early one. The new StE uses similarly an encryption scheme from the KEM-DEM paradigm, and the digital signature is produced on both the encapsulation of the key (used later for encryption) and the message in question.

Theorem 16 *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$, signcryptions from the new StE are (t, ϵ, q_s)-EUF-CMA secure if the underlying digital signature scheme is (t, ϵ, q_s)-EUF-CMA secure.* ∎

Theorem 17 *Given* $(t, q_s, q_v, q_u, q_{cd}, q_{pv}) \in \mathbb{N}^6$ *and* $(\varepsilon, \epsilon') \in [0,1]^2$*, signcryptions from the new StE are* $(t, \epsilon, q_s, q_v, q_u, q_{cd}, q_{pv})$*-IND-CCA secure if they use a* (t, ϵ', q_s)*-EUF-CMA secure digital signature, an IND-OT secure DEM and an* $(t + q_s(q_u + q_{cd} + q_{pv}), \frac{\epsilon}{2} \cdot (1-\epsilon')^{q_u+q_{cd}+q_{pv}})$*-IND-CPA secure KEM.*

Proof (Sketch) As in the proof of Theorem 6, the reduction $\mathcal{R}$ (algorithm trying to IND-CPA break the KEM) simulates the environment of the IND-CCA attacker $\mathcal{A}$ against the construction using the sender's keypair (that she generates internally) and a list of records in which she maintains the queries, the corresponding responses and the intermediate values used to generate these responses.

At some point $\mathcal{A}$ will output two messages m_0 and m_1. $\mathcal{R}$ uses then her challenge, say (c, k), to produce a digital signature σ_b on m_b for $b \xleftarrow{R} \{0,1\}$ (chosen by $\mathcal{R}$). Finally, she produces an encryption e_b (using the DEM encryption algorithm) on $m_b \| \sigma_b$ using k, and outputs $\mu = (c, e_b)$ as a challenge signcryption to $\mathcal{A}$. Note that the challenge (c, k) is computed as follows; if some $b' \xleftarrow{R} \{0,1\}$ is 1, then k is the decapsulation of c, otherwise k is a string chosen uniformly at random from the key space. Thus, if k is the decapsulation of c ($b' = 1$), then μ is a valid signcryption on m_b. Otherwise, it is not a valid signcryption on either messages.

If the advantage of $\mathcal{A}$ is non-negligibly different from the advantage of an indistinguishability adversary in a real attack, i.e. $\mathcal{A}$ responds with $1-b$ with high probability when k is not the decapsulation of c (to denote that μ is not a signcryption of m_b), then $\mathcal{A}$ can be easily turned into an attacker against the IND-OT security property of the DEM underlying the construction. In fact, when $\mathcal{A}$ hands the messages m_0, m_1 to $\mathcal{R}$ (adversary against the IND-OT property of the DEM), this latter produces a pair $(k, c) = \mathcal{K}.\mathtt{encap}()$ consisting of a key and of its encapsulation, and then gives $m_0 \| \Sigma.\mathtt{sign}(m_0 \| c)$ and $m_1 \| \Sigma.\mathtt{sign}(m_1 \| c)$ as challenge messages to her IND-OT challenger. Upon receipt of the challenge encryption e (using a key different from k) of $m_b \| \Sigma.\mathtt{sign}(m_b \| c)$ for some $b \xleftarrow{R} \{0,1\}$, $\mathcal{R}$ will forward (c, e) to $\mathcal{A}$. When this latter answers with b', then $\mathcal{R}$ will respond with $1 - b'$.

To sum up, under the IND-OT assumption of the DEM underlying the construction, the challenge μ is compatible with the indistinguishability game for signcryption.

Finally, when $\mathcal{A}$ outputs his guess b_a, $\mathcal{R}$ will output $b_r = 1$ to her own IND-CPA challenger if $b = b_a$ and 0 otherwise.

Theorem 18 *Given* $(t, q_s) \in \mathbb{N}^2$ *and* $(\varepsilon, \epsilon_b) \in [0,1]^2$*, signcryptions from the new CtEaS are* (t, ϵ, q_s)*-EUF-CMA secure if they use a* (t, ϵ_b) *binding commitment scheme and a* $(t, \epsilon(1-\epsilon_b)^{q_s}, q_s)$*-EUF-CMA secure digital signature scheme.* ∎

Theorem 19 *Given* $(t, q_s, q_v, q_u, q_{cd}, q_{pv}) \in \mathbb{N}^6$ *and* $(\varepsilon, \epsilon', \epsilon_h) \in [0,1]^3$*, signcryptions from the new CtEaS are* $(t, \epsilon, q_s, q_v, q_u, q_{cd}, q_{pv})$*-IND-CCA secure if they use a* (t, ϵ', q_s)*-SEUF-CMA secure digital signature, a statistically binding, and* (t, ϵ_h)*-hiding commitment, and a* $(t + q_s(q_u + q_{cd} + q_{pv}), \frac{1}{2}(\epsilon + \epsilon_h)(1-\epsilon')^{q_u+q_{cd}+q_{pv}})$*-IND-CPA secure encryption scheme.* ∎

7.5 A new paradigm for efficient verifiable signcryption

Like in confirmer signatures, EtS for signcryption compares better with respect to verifiability, since the sender needs simply to prove knowledge of the decryption of a given ciphertext. Also, the receiver has to prove that a message is/isn't the decryption of a given ciphertext. Such proofs are easy to carry out if one considers the already mentioned class $\mathbb{E}$. Moreover, `publicVerify` (similar to conversion in confirmer signatures) can be made efficient for many encryption schemes from the class $\mathbb{E}$. However, in order to achieve indistinguishability, EtS exacts that the underlying signature satisfies the highest security notion, i.e. strong unforgeability under chosen message attacks. Such a need is justified by the possibility, in case the signature scheme does not satisfy this requirement, to create a new signcryption on any message given one signcryption on it (just generate a new digital signature on the encryption of the message), which entitles the indistinguishability adversary to retrieve the message in the standard indistinguishability game. Besides, signcryptions from EtS are not anonymous, i.e. disclose the identity of the sender (anyone can check the validity of the digital signature on the ciphertext w.r.t. the sender's public key).

The new StE paradigm, described in the previous subsection, does not suffer the recourse to stronger assumptions on the underlying signature. It further provides anonymity of the sender if the used encryption is anonymous (the signcryption on a message m is a ciphertext). However, verifiability turns out to be a hurdle; StE applies the signing algorithm (of the used signature scheme) to the message to be signcrypted concatenated with the used encapsulation. It further produces an encryption of the resulting signature concatenated with the message in question. As we are interested in proving the validity of the produced signcryption, we will need to exploit the homomorphic properties of the signature and of the encryption schemes in order to provide proofs of knowledge of the encrypted signature and message. As a consequence, the used encryption and signature schemes need to operate on elements from a set with a known algebraic structure rather than on bit-strings. The same thing applies to the new CtEaS paradigm as the encryption is performed on the concatenation of the message to be signcrypted and the opening value of the commitment scheme.

To sum-up, EtS provides efficient verifiability but at the expense of the sender's anonymity, and of the security requirements on the building blocks. StE achieves better privacy using cheap constituents but at the expense of verifiability. It would be nice to have a technique that combines the merits of both paradigms while avoiding their drawbacks. This is the main contribution in this subsection; the core of the idea consists in first encrypting the message to be signcrypted using a public key encryption scheme, then applying StE to the produced encryption. The result of this operation in addition to the encrypted message form the new signcryption of the message in question. In other terms, this technique can be seen as a merge between EtS and StE; thus we can term it the "encrypt-then-sign-then-encrypt" paradigm (EtStE).

7.5.1 Construction

Consider a signature scheme Σ, an encryption scheme Γ, and another encryption scheme $(\mathcal{K}, \mathcal{D})$ derived from the KEM/DEM paradigm. Next, on input the security parameter κ, generate the parameters *param* of these schemes. We assume that signatures issued with Σ can be written as (r, s), where r reveals no information about the signed message nor about the public signing key, and s represents the "significant" part of the signature. Signcryptions from EtStE are as follows:

Key generation. Invoke the key generation algorithms of the building blocks and set the sender's key pair to $(\Sigma.pk, \Sigma.sk)$, and the receiver's key pair to $(\{\Gamma.pk, \mathcal{K}.pk\}, \{\Gamma.sk, \mathcal{K}.sk\})$.

Signcrypt. On a message m, produce an encryption $e = \Gamma.\mathtt{encrypt}_{\Gamma.pk}(m)$ of m. Then fix a key k along with its encapsulation c using $\mathcal{K}.\mathtt{encrypt}_{\mathcal{K}.pk}$, produce a signature (r, s) on $c\|e$, and finally encrypt s with k using $\mathcal{D}.\mathtt{encrypt}$. The signcryption of m is the tuple $(e, c, \mathcal{D}.\mathtt{encrypt}_k(s), r)$.

Prove Validity. Given a signcryption $\mu = (\mu_1, \mu_2, \mu_3, \mu_4)$ on a message m, the prover proves knowledge of the decryption of μ_1, and of the decryption of (μ_2, μ_3), which together with μ_4 forms a valid digital signature on $\mu_2\|\mu_1$. The private input is either the randomness used to create μ or $\{\Gamma.sk, \mathcal{K}.sk\}$.

Unsigncrypt. On a signcryption a $(\mu_1, \mu_2, \mu_3, \mu_4)$, compute $m = \Gamma.\mathtt{decrypt}_{\Gamma.sk}(\mu_1)$ and $k = \mathcal{K}._{\mathcal{K}.sk}(\mu_2)$. Check whether $(\mathcal{D}.\mathtt{decrypt}_k(\mu_3), \mu_4)$ is valid signature on $\mu_2\|\mu_1$; if yes then output m, otherwise output $\perp$.

Confirm/Deny. On input a putative signcryption $\mu = (\mu_1, \mu_2, \mu_3, \mu_4)$ on a message m, use the receiver's private key to prove that m is/isn't the decryption of μ_1, and prove knowledge of the decryption of (μ_2, μ_3), which together with μ_4 forms a valid/invalid digital signature on $\mu_2\|\mu_1$.

Public Verify. On a valid signcryption $\mu = (\mu_1, \mu_2, \mu_3, \mu_4)$ on a message m, output a ZK non-interactive proof that μ_1 encrypts m, in addition to $(\mathcal{D}.\mathtt{decrypt}_{\mathcal{K}.\mathtt{decap}(\mu_2)}(\mu_3), \mu_4)$.

Theorem 20 *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, the above construction is (t, ϵ, q_s)-EUF-CMA secure if the underlying digital signature scheme is (t, ϵ, q_s)-EUF-CMA secure.* ∎

Theorem 21 *Given $(t, q_s, q_v, q_u, q_{cd}, q_{pv}) \in \mathbb{N}^6$ and $(\varepsilon, \epsilon') \in [0, 1]^2$, the construction proposed above is $(t, \epsilon, q_s, q_v, q_u, q_{cd}, q_{pv})$-IND-CCA secure if it uses a (t, ϵ', q_s)-EUF-CMA secure digital signature, an IND-CPA secure encryption, an IND-OT secure DEM, and a $(t + q_s(q_u + q_{cd} + q_{pv}), \frac{\epsilon(1-\epsilon')^{q_{cd}+q_u+q_{pv}}}{2})$-IND-CPA secure KEM.*

Proof (Sketch) From an IND-CCA adversary $\mathcal{A}$ against the construction, we construct an algorithm $\mathcal{R}$ that IND-CPA break the KEM underlying the construction. $\mathcal{R}$ gets the public parameters of the KEM from her challenger and chooses further the remaining building blocks, i.e. the DEM, the signature, and the encryption scheme. Simulation of $\mathcal{A}$'s environment is done using the key pairs of the used signature and encryption schemes, in addition to a list in which $\mathcal{R}$ maintains the queries, their responses and the intermediate values used to generate these responses.

Eventually, $\mathcal{A}$ outputs two challenge messages m_0, m_1. $\mathcal{R}$ will encrypt, in e_b, the message m_b, for $b \xleftarrow{R} \{0,1\}$. Next, she produces a signature (r,s) on $c\|e_b$, where (c,k) is her challenge. Finally, $\mathcal{R}$ encrypts s in $e_{\mathcal{D}}$ using k, and outputs $\mu = (e_b, c, e_{\mathcal{D}}, r)$ as a challenge signcryption. Since the used encryption is IND-CPA secure by assumption, then information about m_b can only leak from $(c, e_{\mathcal{D}}, r)$. If k is the decapsulation of c, then μ is a valid signcryption of m_b, otherwise it is not a valid signcryption on either messages. The rest of the proof follows in a straightforward way. ∎

7.5.2 Instantiations

The `proveValidity` and {`confirm`, `deny`} protocols comprise the following sub-protocols:

1. Proving knowledge of the decryption of a ciphertext produced using the encryption scheme Γ.

2. Proving that a message is/isn't the decryption of a certain ciphertext produced using Γ.

3. Proving knowledge of the decryption of a ciphertext produced using $(\mathcal{K}, \mathcal{D})$, and that this decryption forms a valid/invalid digital signature, issued using Σ, on some known string.

It is natural to instantiate the encryption scheme Γ from the class $\mathbb{E}$ described in Definition 5. With this choice, the first two sub-protocols can be efficiently carried out as depicted in Figure 4 and Figure 7 respectively. Moreover, one can consider encryption schemes from the class $\mathbb{E}$ that are derived from the KEM/DEM paradigm, in addition to signature schemes from the class $\mathbb{S}$ described in Definition 4. The last sub-protocol boils down then to the protocol depicted in Figure 6.

Finally, for the `publicVerify` algorithm, we refer to the solutions adopted in confirmer signatures (described in Paragraph 6.7.2) when it comes to producing a NIZK proof of the correctness of a decryption.

7.6 Extension to multi-user signcryption

So far, we considered signcryption schemes in the two-user setting, i.e. a single sender interacts with a single receiver. A signcryption scheme secure in the two-user setting does not necessarily mean that it conserves this security in the multi-user setting. In fact, the unforgeability adversary in the latter mode is allowed to return a forgery on a message $m^\star$ that may have been queried before but w.r.t. a receiver's key different from the target receiver's key $pk_{\mathsf{R}}^\star$. Moreover, the indistinguishability adversary is allowed to ask the unsigncryption or (public) verification of the challenge w.r.t. any receiver's key except that of the target receiver. However many works [1, 52] have proposed simple tweaks in order to derive multi-user security from two-user security. These techniques apply also for our constructions in order to guarantee security in the multi-user setting.

For instance, the EtS paradigm in the multi-user setting departs from that of the two-user setting in the following elements:

1. It considers a tag-based encryption scheme where the tag is set to the public key of the sender pk_{S}.

2. The digital signature is produced on the resulting ciphertext and on the public key of the receiver.

Similarly, the EtStE paradigm in the multi-user setting deviates from that of the two-user one as follows:

1. It considers a tag-based KEM where the tag is set to the public key of the sender pk_{S}.

2. The digital signature is produced on the resulting ciphertext and on the public key of the receiver.

8 Verifiably Encrypted Signatures (VES)

8.1 Syntax and model

A verifiably encrypted signature scheme comprise the following algorithms:

`keygen`, `sign`, **and** `verify` as in standard signature schemes.

Experiment $\mathbf{Exp}^{\text{unforgeability}}(1^\kappa)$	Experiment $\mathbf{Exp}^{\text{opacity}}(1^\kappa)$
1. $param \leftarrow \texttt{setup}(1^\kappa)$; 2. $(pk, sk) \leftarrow \texttt{keygen}(1^\kappa)$; 3. $(pk_{\mathsf{A}}, sk_{\mathsf{A}}) \leftarrow \texttt{keygen}_{1^\kappa)}$; 4. $(m^\star, \mu^\star) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{A}}(pk, pk_{\mathsf{A}})$ $\mathfrak{S} : m \longmapsto \texttt{VESign}_{\{sk,pk_{\mathsf{A}}\}}(m)$ $\mathfrak{A} : (\mu, m) \longmapsto \texttt{adjudicate}_{\{sk_{\mathsf{A}},pk_{\mathsf{A}},pk\}}(\mu, m)$ 5. return 1 if and only if: - $\texttt{VESverify}_{\{pk,pk_{\mathsf{A}}\}}[m^\star, \mu^\star] = 1$ - $m^\star$ was not queried to $\mathfrak{S}$ and $\mathfrak{A}$.	1. $param \leftarrow \texttt{setup}(1^\kappa)$; 2. $(pk, sk) \leftarrow \texttt{keygen}(1^\kappa)$; 3. $(pk_{\mathsf{A}}, sk_{\mathsf{A}}) \leftarrow \texttt{keygen}_{1^\kappa)}$; 4. $(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{A}}(pk, pk_{\mathsf{A}})$ $\mathfrak{S} : m \longmapsto \texttt{VESign}_{\{sk,pk_{\mathsf{A}}\}}(m)$ $\mathfrak{A} : (\mu, m) \longmapsto \texttt{adjudicate}_{\{sk_{\mathsf{A}},pk_{\mathsf{A}},pk\}}(\mu, m)$ 5. return 1 if and only if: - $\texttt{verify}_{\{pk\}}[m^\star, \sigma^\star] = 1$ - $m^\star$ was not queried to $\mathfrak{A}$.

Figure 9: Security in verifiably encrypted signatures

$\texttt{keygen}_{\mathsf{A}}(1^\kappa, param)$ generates a key pair $(sk_{\mathsf{A}}, pk_{\mathsf{A}})$ for the adjudicator.

$\texttt{VESign}(m, sk, pk_{\mathsf{A}})$ on input the signer's private key sk, a message m, and an ajudicator's public key pk_{A} output (probabilistically) a verifiably encrypted signature μ.

$\texttt{VESverify}(m, \mu, pk_{\mathsf{A}}, pk_{\mathsf{S}})$ on input the signer's public key pk, a message m, an adjudicator's public key pk_{A}, and a verifiably encrypted signature μ, output 1 if μ is a valid verifiably encrypted signature on m under pk, and 0 otherwise.

$\texttt{adjudicate}(m, \mu, sk_{\mathsf{A}}, pk_{\mathsf{A}}, pk)$ on input the adjudicator key pair $(pk_{\mathsf{A}}, sk_{\mathsf{A}})$, a certified public key pk, and a verifiably encrypted signature μ on a message m, extract and output an ordinary signature on m under pk.

A verifiably encrypted scheme should satisfy three security properties, namely *correctness*, *unforgeability*, and *opacity*.

Correctness requires that:

$$\texttt{VESverify}_{\{pk,pk_{\mathsf{A}}\}}(\texttt{VESign}_{sk,pk_{\mathsf{A}}}(m), m) = 1,$$

and

$$\texttt{verify}_{pk}(\texttt{adjudicate}_{\{sk_{\mathsf{A}},pk_{\mathsf{A}},pk\}}[\texttt{VESign}_{sk,pk_{\mathsf{A}}}(m), m], m) = 1$$

for all messages m, and properly generated keys (sk, pk) and $(sk_{\mathsf{A}}, pk_{\mathsf{A}})$.

Unforgeability requires the difficulty to forge a valid verifiably encrypted signature. The adversary, say $\mathcal{A}$, is similarly given access to a VESign and a adjudicate oracles that he can query for any input of his choice. At the end $\mathcal{A}$ must output a message $m^\star$, that has not been queried to either oracles, along with a valid verifiably encrypted signature $\mu^\star$ on it.
We say that a VES is (t, ϵ, q_s, q_a)-EUF-CMA secure if there is no adversary, operating in time t, and issuing q_s queries to the VESign oracle and q_a queries to the adjudicate oracle, that wins the unforgeability game (depicted in Figure 9) with probability greater than ϵ, where the probability is taken over the random choices of both $\mathcal{A}$ and his challenger.

Finally, opacity requires the difficulty to extract from a verifiably encrypted signature an ordinary signature on the same message. Similarly, the adversary $\mathcal{A}$ is allowed to query the signing and adjudication oracles on any input. At the end, $\mathcal{A}$ should return an ordinary signature on some message $m^\star$, that has not been queried previously to the adjudication oracle.
We say that a VES is (t, ϵ, q_s, q_a)-opaque if there is no adversary, operating in time t, and issuing q_s queries to the VESign oracle and q_a queries to the adjudicate oracle, that wins the opacity game (depicted in Figure 9) with probability greater than ϵ, where the probability is taken over the random choices of both $\mathcal{A}$ and his challenger.

8.2 Analysis of the StE paradigm

Verifiably encrypted signatures can be achieved as the name suggests by first producing an ordinary signature on the message to be signed, then encrypting it (the signature) under the adjudicator's public key. The result forms the verifiably encrypted signature on the message. To ensure the verifiability of the result, one can proceed to two solutions. The first consists in augmenting the VES with a non-interactive proof of validity, i.e. proving that the VES encrypts a valid digital signature on the message in question. However, since NIZK are in general difficult to achieve, we won't elaborate this solution further. The second solution generalizes what is deployed in the literature. It considers special classes of signature and of encryption schemes; the class of signatures generalizes the class $\mathbb{S}$ described earlier in this document in the sense that it does not require the homomorphic property on the function f. The class of encryption includes schemes where any pair of message and corresponding ciphertext, under a given key, satisfies a relation confined by the function f. More specifically, these classes are defined as follows.

Definition 9 *$\mathbb{S}$ is the set of all digital signatures for which there exists a pair of efficient algorithms,* `convert` *and* `retrieve`*, where* `convert` *inputs a public key pk, a message m, and a valid signature σ on m (according to pk) and outputs the pair (s, r) such that:*

1. *r reveals no information about m nor about pk.*
2. *there exists an algorithm* `compute` *that on the input pk, the message m and r, computes a description of a function f and an $I \in \mathbb{H}$, such that $f(s) = I$.*

and `retrieve` *is an algorithm that inputs pk, m and the correctly converted pair (s, r) and retrieves the signature σ on m.*

Definition 10 *$\mathbb{E}$ is the set of all public key encryption scheme for which there exists efficient maps g and h for every valid public key pk such that:*

1. *for any message m and corresponding ciphertext c under pk, we have: $h(c) = g(m)$.*
2. *from any pair of message m and ciphertext c, if $h(c) = g(m)$, then c is encryption of m under pk.*

8.2.1 Construction

Let Σ be a signature scheme from $\mathbb{S}$ and let further Γ be an encryption scheme from $\mathbb{E}$, where the map g (in Definition 10) is the composition of some efficient map f' and the map f (in Definition 9): $g = f' \circ f$. We combine now Σ and Γ using StE, namely we produce a signature σ on m that we convert into (s, r), then we encrypt s in c and we finally output $\mu = (c, r)$ as a VES on m. It is clear that one can verify the correctness of this VES by simply checking whether $h(c) = f'(I)$, where I is computed according to Definition 9. An illustration of VES from this construction is the celebrated Boneh *et al.*'s verifiably encrypted signature [8] where the underlying signature is BLS's [9] signature (the converted pair from a BLS signature σ is (σ, ϵ), and $f(\sigma) = \langle \sigma, P \rangle$), and the underlying encryption is ElGamal's [34] encryption scheme ($h(P^r, mY^r) = \langle mY^r, P \rangle \langle Y, P^r \rangle^{-1}$, and $g(m) = \langle m, P \rangle$).

It is clear that an encryption scheme from $\mathbb{E}$ cannot be NM-CPA nor IND-CPA secure due to the maps h and g which allow to efficiently check whether a ciphertext encrypts a given message under some given key. We will now show that the opacity of a VES from the above construction cannot be reduced to the OW-CCA security of the underlying encryption. Similarly, we show this result for key-preserving reductions, and the generalization to arbitrary reductions can be achieved as presented in Subsection 4.3.

Lemma 7 *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an opacity adversary $\mathcal{A}$ against VES from the StE paradigm to a OW-CCA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that OW-CCA breaks the encryption scheme in question.*

Proof Let $\mathcal{R}$ be the key-preserving reduction that reduces OW-CCA breaking the encryption scheme underlying the construction to breaking the opacity of the construction. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to OW-CCA break the same encryption scheme.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack. $\mathcal{M}$ gets his challenge c (along with a public key pk) and is equipped with a decryption oracle that he can query on all ciphertexts except c. $\mathcal{M}$ launches

$\mathcal{R}$ over Γ with the same public key pk and the same challenge c. Obviously, all decryption queries made by $\mathcal{R}$, which are by definition different from c, can be forwarded to $\mathcal{M}$'s own challenger. $\mathcal{M}$ needs now to simulate the opacity adversary to $\mathcal{R}$. Thus, $\mathcal{M}$ queries $\mathcal{R}$ for a VES on some message m. Let μ be the response of $\mathcal{R}$ to such a query. Note that $\mathcal{M}$ is able to check the validity of this VES w.r.t. m. Two cases, either $\mu = c$ (very improbable) in which $\mathcal{M}$ can ask $\mathcal{R}$ for the adjudication of (c, m). The result of such a query allows $\mathcal{M}$ to answer his OW-CCA challenge. If $\mu \neq c$, then $\mathcal{M}$ can query his own challenger for the decryption of μ. The response to such a query, along with m, form the opacity forgery that $\mathcal{M}$ will output to $\mathcal{R}$. Thus, $\mathcal{R}$ is expected to answer the OW-CCA challenge, namely the decryption of c. Upon receipt of this answer, $\mathcal{M}$ will forward it to his own challenger. ∎

This negative result illustrates that the insufficiency of a security notion (e.g. OW-CCA) proven using the above tool, i.e. meta-reductions, does not necessarily rule out a weaker notion (e.g. OW-PCA) if there are separations between those notions. This is the case for ElGamal's encryption in bilinear groups.

9 Conclusion

In this paper, we studied the classical paradigms used to build many opaque signatures, namely StE, EtS, and CtEtS. We showed using an increasingly popular tool, namely meta-reductions, that StE and CtEaS require expensive encryption in order to provide a reasonable security level for the resulting construction. This is due to an intrinsic weakness of those paradigms which consists in the possibility of obtaining the opaque signature without the help of the signer. Next, we proposed some adjustments to these paradigms which circumvent this weakness and allow to rest on cheap encryption without compromising the security level of the result. We further gave many practical instantiations of these paradigms which efficiently implement the verifiability feature in the constructions, i.e. the possibility to prove the validity of the opaque signature.

The immediate prospect of such an analysis is its extension to other opaque or privacy-preserving mechanisms/signatures, e.g. group signatures or anonymous credentials. In fact, most such mechanisms involve a digital signature on some message and an encryption layer that ensures the privacy. Hence the possibility of applying the same techniques in order to allow cheap and useful encryption in the design, and thus achieve constructions with many efficient instantiations. The long-run prospect consists in systematically applying the meta-reduction tool in other cryptographic realizations in order to spot the potential flaws in the design, and later repair these flaws and improve the resulting constructions.

References

[1] J. H. An, Y. Dodis, and T. Rabin, *On the Security of Joint Signature and Encryption.*, Advances in Cryptology - Eurocrypt 2002 (L. R. Knudsen, ed.), LNCS, vol. 2332, Springer, 2002, pp. 83–107.

[2] J. Baek, R. Steinfeld, and Y. Zheng, *Formal Proofs for the Security of Signcryption*, J. Cryptology **20** (2007), no. 2, 203–235.

[3] 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.

[4] M. Bellare and P. Rogaway, *The Exact Security of Digital Signatures: How to Sign with RSA and Rabin.*, in Maurer [53], pp. 399–416.

[5] M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)*, STOC (J. Simon, ed.), ACM Press, 1988, pp. 103–112.

[6] D. Boneh and X. Boyen, *Short Signatures Without Random Oracles.*, Advances in Cryptology - Eurocrypt 2004 (C. Cachin and J. Camenisch, eds.), LNCS, vol. 3027, Springer, 2004, pp. 56–73.

[7] D. Boneh, X. Boyen, and H. Shacham, *Short Group Signatures.*, in Franklin [35], pp. 41–55.

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps.*, Advances in Cryptology - Eurocrypt 2003 (E. Biham, ed.), LNCS, vol. 2656, Springer, 2003, pp. 416–432.

[9] D. Boneh, B. Lynn, and H. Shacham, *Short Signatures from the Weil Pairing.*, J. Cryptology **17** (2004), no. 4, 297–319.

[10] D. Boneh and R. Venkatesan, *Breaking RSA May Not Be Equivalent to Factoring.*, Advances in Cryptology - Eurocrypt'98 (K. Nyberg, ed.), LNCS, vol. 1403, Springer, 1998, pp. 59–71.

[11] C. Boyd and E. Foo, *Off-line Fair Payment Protocols using Convertible Signatures.*, Advances in Cryptology - Asiacrypt'98 (K. Ohta and D. Pei, eds.), LNCS, vol. 1514, Springer, 1998, pp. 271–285.

[12] G. Brassard, D. Chaum, and C. Crépeau, *Minimum disclosure proofs of knowledge.*, J. Comput. Syst. Sci. **37** (1988), no. 2, 156–189.

[13] J. Camenisch and A. Lysyanskaya, *Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials*, CRYPTO (M. Yung, ed.), LNCS, vol. 2442, Springer, 2002, pp. 61–76.

[14] ———, *Signature Schemes and Anonymous Credentials from Bilinear Maps*, in Franklin [35], pp. 56–72.

[15] J. Camenisch and M. Michels, *Confirmer Signature Schemes Secure against Adaptative Adversaries.*, in Preneel [60], pp. 243–258.

[16] 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.

[17] D. Chaum, *Designated Confirmer Signatures.*, Advances in Cryptology - Eurocrypt'94 (A. De Santis, ed.), LNCS, vol. 950, Springer, 1995, pp. 86–91.

[18] D. Chaum and H. van Antwerpen, *Undeniable Signatures.*, Advances in Cryptology - Crypto'89 (G. Brassard, ed.), LNCS, vol. 435, Springer, 1990, pp. 212–216.

[19] D. Chiba, T. Matsuda, J. C. N. Schuldt, and K. Matsuura, *Efficient Generic Constructions of Signcryption with Insider Security in the Multi-user Setting*, ACNS (J. Lopez and G. Tsudik, eds.), LNCS, vol. 6715, 2011, pp. 220–237.

[20] S. S. M. Chow and K. Haralambiev, *Non-interactive Confirmer Signatures*, CT-RSA (A. Kiayias, ed.), LNCS, vol. 6558, Springer, 2011, pp. 49–64.

[21] R. Cramer (ed.), *Public key cryptography - pkc 2008, 11th international workshop on practice and theory in public-key cryptography, barcelona, spain, march 9-12, 2008. proceedings*, LNCS, vol. 4939, Springer, 2008.

[22] R. Cramer and V. Shoup, *Signature schemes based on the strong RSA assumption.*, ACM Trans. Inf. Syst. Secur. **3** (2000), no. 3, 161–185.

[23] ———, *Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack.*, SIAM J. Comput. **33** (2003), no. 1, 167–226.

[24] I. B. Damgård and T. P. Pedersen, *New Convertible Undeniable Signature Schemes.*, in Maurer [53], pp. 372–386.

[25] I. Damgård, *Efficient concurrent zero-knowledge in the auxiliary string model*, in Preneel [60], pp. 418–430.

[26] I. Damgård, N. Fazio, and A. Nicolosi, *Non-interactive zero-knowledge from homomorphic encryption*, in Halevi and Rabin [45], pp. 41–59.

[27] A. W. Dent, *Hybrid Signcryption Schemes with Outsider Security*, ISC (J. Zhou, J. Lopez, R. H. Deng, and F. Bao, eds.), LNCS, vol. 3650, Springer, 2005, pp. 203–217.

[28] C Dwork, M. Naor, and A. Sahai, *Concurrent Zero-Knowledge.*, J. Assoc. Comput. Mach. **51** (2004), no. 6, 851–898.

[29] L. El Aimani, *Toward a Generic Construction of Universally Convertible Undeniable Signatures from Pairing-Based Signatures*, Progress in Cryptology - Indocrypt 2008 (D. Roy Chowdhury, V. Rijmen, and A. Das, eds.), LNCS, vol. 5365, Springer, 2008, Full version available at the Cryptology ePrint Archive, Report 2009/362, pp. 145–157.

[30] ______, *Anonymity from Public Key Encryption to Undeniable Signatures*, Africacrypt 2009 (B. Preneel, ed.), LNCS, vol. 5580, Springer, 2009, pp. 217–234.

[31] ______, *On Generic Constructions of Designated Confirmer Signatures*, in Roy and Sendrier [62], Full version available at the Cryptology ePrint Archive, Report 2009/403, pp. 343–362.

[32] ______, *Efficient Confirmer Signature from the "Signature of a Commitment" Paradigm*, ProvSec 2010 (S-H. Heng and K. Kurosawa, eds.), LNCS, vol. 6402, Springer, 2010, Full version available at the Cryptology ePrint Archive, Report 2009/435, pp. 87–101.

[33] ______, *Generic Constructions for Verifiable Signcryption*, ICISC'11 (H. Kim, ed.), LNCS, Springer, 2011, Available at the Cryptology ePrint Archive. Report 2011/592, p. To appear.

[34] T. El Gamal, *A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms.*, IEEE Trans. Inf. Theory **31** (1985), 469–472.

[35] M. K. Franklin (ed.), *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, LNCS, vol. 3152, Springer, 2004.

[36] 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.

[37] R. Gennaro, S. Halevi, and T. Rabin, *Secure Hash-and-Sign Signatures Without the Random Oracle.*, in Stern [68], pp. 397–416.

[38] C. Gentry, D. Molnar, and Z. Ramzan, *Efficient Designated Confirmer Signatures Without Random Oracles or General Zero-Knowledge Proofs*, in Roy [61], pp. 662–681.

[39] O. Goldreich, *Foundations of cryptography. Basic Tools.*, Cambridge University Press., 2001.

[40] Oded Goldreich, Silvio Micali, and Avi Wigderson, *How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design*, CRYPTO (A. M. Odlyzko, ed.), LNCS, vol. 263, Springer, 1986, pp. 171–185.

[41] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems.*, SIAM J. Comput. **18** (1989), no. 1, 186–206.

[42] 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.

[43] 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.

[44] 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.

[45] Shai Halevi and Tal Rabin (eds.), *Theory of cryptography, third theory of cryptography conference, tcc 2006, new york, ny, usa, march 4-7, 2006, proceedings*, LNCS, vol. 3876, Springer, 2006.

[46] J. Herranz, D. Hofheinz, and E. Kiltz, *KEM/DEM: Necessary and Sufficient Conditions for secure Hybrid Encryption*, Available at http://eprint.iacr.org/2006/265.pdf, August 2006.

[47] M. Jakobsson, K. Sako, and R. Impagliazzo, *Designated Verifier Proofs and Their Applications.*, in Maurer [53], pp. 143–154.

[48] I. Jeong, H. Jeong, H. Rhee, D. Lee, and J. Lim, *Provably Secure Encrypt-then-Sign Composition in Hybrid Signcryption*, ICISC (P. J. Lee and C. H. Lim, eds.), LNCS, vol. 2587, Springer, 2002, pp. 16–34.

[49] E. Kiltz, *Chosen-Ciphertext Security from Tag-Based Encryption*, in Halevi and Rabin [45], pp. 581–600.

[50] P. Le Trieu, K. Kurosawa, and W. Ogata, *Provably Secure Convertible Undeniable Signatures with Unambiguity*, SCN 2010 (J A. Garay and R. De Prisco, eds.), LNCS, vol. 6480, Springer, 2010, Full version available at the Cryptology ePrint Archive, Report 2009/394.

[51] M. Liskov and S. Micali, *Online-Untransferable Signatures*, in Cramer [21], pp. 248–267.

[52] T. Matsuda, K. Matsuura, and J. Schuldt, *Efficient Constructions of Signcryption Schemes and Signcryption Composability*, in Roy and Sendrier [62], pp. 321–342.

[53] 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.

[54] J. Monnerat and S. Vaudenay, *Short Undeniable Signatures Based on Group Homomorphisms*, J. Cryptology **24** (2011), no. 3, 545–587.

[55] P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, in Stern [68], pp. 223–238.

[56] ———, *Impossibility Proofs for RSA Signatures in the Standard Model*, CT-RSA (M. Abe, ed.), LNCS, vol. 4377, Springer, 2007, pp. 31–48.

[57] P. Paillier and D. Vergnaud, *Discrete-Log Based Signatures May Not Be Equivalent to Discrete-Log.*, in Roy [61], pp. 1–20.

[58] 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.

[59] D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures.*, J. Cryptology **13** (2000), no. 3, 361–396.

[60] B. Preneel (ed.), *Advances in Cryptology* - Eurocrypt *2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, LNCS, vol. 1807, Springer, 2000.

[61] 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.

[62] B. Roy and N. Sendrier (eds.), *Progress in cryptology* - Indocrypt *2009*, vol. 5922, Berlin, Heidelberg, 2009.

[63] M. Rückert and Schröder D., *Aggregate and Verifiably Encrypted Signatures from Multilinear Maps without Random Oracles*, ISA (J. Park, H. Chen, M. A., C. Lee, T-H. Kim, and S-S. Yeo, eds.), LNCS, vol. 5576, Springer, 2009, pp. 750–759.

[64] C. P. Schnorr, *Efficient signature generation by smart cards.*, J. Cryptology **4** (1991), no. 3, 161–174.

[65] J. C. N. Schuldt and K. Matsuura, *An Efficient Convertible Undeniable Signature Scheme with Delegatable Verification*, ISPEC 2010 (J. Kwak, R. H. Deng, Y. Won, and G. Wang, eds.), LNCS, vol. 6047, Springer, 2010, Full version available at the Cryptology ePrint Archive, Report 2009/454, pp. 276–293.

[66] S. F. Shahandashti and R. Safavi-Naini, *Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures*, in Cramer [21], pp. 121–140.

[67] V. Shoup and R. Gennaro, *Securing Threshold Cryptosystems against Chosen Ciphertext Attack*, J. Cryptology **15** (2002), no. 2, 75–96.

[68] J. Stern (ed.), *Advances in Cryptology* - Eurocrypt*'99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, LNCS, vol. 1592, Springer, 1999.

[69] 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.

[70] B. Waters, *Efficient Identity-Based Encryption Without Random Oracles.*, Advances in Cryptology - Eurocrypt 2005 (R. Cramer, ed.), LNCS, vol. 3494, Springer, 2005, pp. 114–127.

[71] D. Wikström, *Designated Confirmer Signatures Revisited*, TCC 2007 (S. P. Vadhan, ed.), LNCS, vol. 4392, Springer, 2007, pp. 342–361.

[72] F. Zhang, R. Safavi-Naini, and W. Susilo, *An Efficient Signature Scheme from Bilinear Pairings and Its Applications.*, 7th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2004 (F. Bao, R. H. Deng, and J. Zhou, eds.), LNCS, vol. 2947, Springer, 2004, pp. 277–290.

[73] Y. Zheng, *Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption).*, Advances in Cryptology - Crypto'97 (B. S. Kaliski Jr., ed.), LNCS, vol. 1294, Springer, 1997, pp. 165–179.

[74] Y. Zhou and H. Qian, *Efficient CDH-Based Verifiably Encrypted Signatures with Optimal Bandwidth in the Standard Model*, ADHOC-NOW, 2011, pp. 164–176.

A Analysis of Damgård-Pedersen's [24] undeniable signatures

A.1 The scheme

Let $m \in \{0,1\}^\star$ be an arbitrary message, the scheme consists of the following procedures:

Setup (`setup`). On input the security parameter κ, generate a k-bit prime t and a prime $p \equiv 1 \bmod t$. Furthermore, select a collision-resistant hash function H that maps arbitrary-length messages to $\mathbb{Z}_t$.

Key generation (`keygen`). Generate g of order t, $x \in \mathbb{Z}_t^\times$, and $h = g^x \bmod p$. Furthermore, select a generator α of $\mathbb{Z}_t^\times$ and $\nu \in \{0,1,\ldots,t-1\}$, and compute $\beta = \alpha^\nu \bmod t$. The public key is $pk = (p,t,g,h,\alpha,\beta)$ and the private key is (x,ν).

Signature (`sign`). The signer first computes an ElGamal signature (s,r) on m, i.e. compute $r = g^b \bmod p$ for some $b \xleftarrow{R} \mathbb{Z}_t^\times$, then compute s as $h(m) = rx + bs \bmod t$. Next, he computes an ElGamal encryption $(E_1 = \alpha^\rho, E_2 = s\beta^\rho) \bmod t$, for $\rho \xleftarrow{R} \mathbb{Z}_{t-1}$, of s. The undeniable signature on m is the triple (E_1,E_2,r).

Confirmation/Denial protocol (`confirm`/`deny`). To confirm (deny) a purported signature (E_1,E_2,r) on a certain message m, the signer issues a ZKPoK of the language: (see [24])

$$\left\{(E_1,E_2,r) \in \mathbb{Z}_t^\times \times \mathbb{Z}_t^\times \times \mathbb{Z}_t^\times \mid \exists\, s \in \mathbb{Z}_t \colon \mathsf{DL}_\alpha(\beta) = \mathsf{DL}_{E_1}(E_2 \cdot s^{-1}) \wedge g^{h(m)}h^{-r} = (\neq) r^s\right\}$$

A.2 Security analysis

Finally, the signatures are proven to be unforgeable if the underlying ElGamal signature is also unforgeable, and they are conjectured (by the authors in [24]) to meet the following security notion if the DDH problem is hard.

Definition 11 (Signature indistinguishability) *It is defined through the following game between an attacker $\mathcal{A}$ (*a distinguisher*) and his challenger $\mathcal{R}$.*

***Phase 1** after $\mathcal{A}$ gets the public parameters of the undeniable signature scheme, namely pk, from $\mathcal{R}$, he starts issuing* status requests *and* signature requests. *In a status request, $\mathcal{A}$ produces a pair (m,z), and receives a 1-bit answer which is 1 iff z is a valid undeniable signature on m w.r.t. pk. In a signature request, $\mathcal{A}$ produces a message m and receives an undeniable signature z on it w.r.t. pk.*

***Challenge** Once $\mathcal{A}$ decides that **Phase 1** is over, he outputs a message m and receives a string z which is either a valid undeniable signature on m (w.r.t pk) or a* simulated signature, *i.e. a string randomly chosen from the signature space.*

***Phase 2** $\mathcal{A}$ resumes adaptively making the previous types of queries, provided that m does not occur in any request, and that z does not occur in any status request. Eventually, $\mathcal{A}$ will output a bit.*

Let p_r, resp. p_s be the probability that $\mathcal{A}$ answers 1 in the real, resp. the simulated case. Both probabilities are taken over the random coins of both $\mathcal{A}$ and $\mathcal{R}$. We say that the signatures are indistinguishable if $|p_r - p_s|$ is a negligible function in the security parameter.

A.3 Negative results

It is clear that the Damgård-Pedersen signatures do not provide the INV-CMA notion according to Subsection 4.1. In the rest of this section, we provide evidence that the Damgård-Pedersen signatures are unlikely to be indistinguishable under the DDH assumption. Indeed, we prove that if there exists a *key-preserving* reduction, i.e. an algorithm launching the adversary over its own public key and other freely chosen parameters, from the DDH problem to the distinguishability of the signatures (in the sense of Definition 11), then there exists an efficient algorithm that solves the DDH problem.

Lemma 8 *Assume there exists a key-preserving reduction $\mathcal{R}$ that uses an indistinguishability adversary $\mathcal{A}$ against the above scheme to solve the DDH problem. Then, there exists an efficient meta-reduction $\mathcal{M}$ that solves the DDH problem.*

Proof Let $\mathcal{R}$ be the key-preserving reduction that reduces the DDH problem to distinguishing the Damgård-Pedersen signatures in the sense of Definition 11. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to solve the DDH problem by simulating a distinguisher against the signatures.

Let $(c_1 = \alpha^a, c_2 = \beta^b) \in \mathbb{Z}_t^\times \times \mathbb{Z}_t^\times$ be the DDH instance $\mathcal{M}$ is asked to solve. $\mathcal{M}$ acting as a distinguisher of the signature will make a signature request on an arbitrary message m. Let (E_1, E_2, r) be the answer to such a query. $\mathcal{M}$ will make now a status query on $(c_1 \cdot E_1, c_2 \cdot E_2, r)$ and the message m. (c_1, c_2) is a yes-Diffie-Hellman instance iff the result of the last query is the confirmation that $(c_1 \cdot E_1, c_2 \cdot E_2, r)$ is a signature on m. ∎

In this case, it does not seem obvious how to extend the above result to arbitrary reductions. For instance, we cannot employ the technique of non-malleability of the key generator used previously in 4.3. In fact, this would correspond in the current case to assume that the DDH problem, w.r.t. a given public key pk, is difficult even when given access to a CDH oracle w.r.t. any $pk' \neq pk$, which is untrue.

Finally, Damgård-Pedersen's undeniable signatures can be repaired so as to provide invisibility by producing ElGamal's signature on the message to be signed concatenated with the used encapsulation E_1. Clearly such a repair is a special instance of our new StE paradigm for CDCS.