Signcryption Schemes with Threshold Unsigncryption, and Applications

Javier Herranz, Alexandre Ruiz, and Germán Sáez

J. Herranz, A. Ruiz and G. Sáez atDept. Matemàtica Aplicada IV, Universitat Politècnica de Catalunya,
C. Jordi Girona, 1-3, Mòdul C3, Barcelona, Spain
{jherranz,aruiz,german}@ma4.upc.edu

Abstract. The goal of a signcryption scheme is to achieve the same functionalities as encryption and signature together, but in a more efficient way than encrypting and signing separately. To increase security and reliability in some applications, the unsigncryption phase can be distributed among a group of users, through a (t, n)-threshold process.
In this work we consider this task of threshold unsigncryption, which has received very few attention from the cryptographic literature up to now (maybe surprisingly, due to its potential applications). First we describe in detail the security requirements that a scheme for such a task should satisfy: existential unforgeability and indistinguishability, under insider chosen message/ciphertext attacks, in a multi-user setting. Then we show that generic constructions of signcryption schemes (by combining encryption and signature schemes) do not offer this level of security in the scenario of threshold unsigncryption. For this reason, we propose two new protocols for threshold unsigncryption, which we prove to be secure, one in the random oracle model and one in the standard model.
The two proposed schemes enjoy an additional property that can be very useful. Namely, the unsigncryption protocol can be divided in two phases: a first one where the authenticity of the ciphertext is verified, maybe by a single party; and a second one where the ciphertext is decrypted by a subset of t receivers, without using the identity of the sender. As a consequence, the schemes can be used in applications requiring some level of anonymity, such as electronic auctions.

Keywords: Signcryption, Threshold cryptography, Electronic auctions.

1 Introduction

By encrypting or signing messages, digital communications may achieve some well-known properties: confidentiality, authentication, integrity or/and non-repudiation. When all these properties are required at the same time, there are more efficient solutions than signing and encrypting each message separately. Cryptographic schemes that provide the same properties than encryption and signature together receive the name of *signcryption* schemes [27] (or also *authenticated encryption* schemes [2]). Such schemes consist of a *key generation* protocol, a *signcryption* protocol run by the sender of the message (which uses his secret key and the public key of the receiver to hide and authenticate the message) and an *unsigncryption* protocol run by the receiver (which uses his secret key and the public key of the sender to recover the message and verify its authenticity).

Since the invention of this concept in 1997, many papers discussing different security properties and proposing new signcryption schemes have appeared. In particular, there are some generic constructions [1] of signcryption schemes, combining signature and encryption schemes, that achieve a very high level of security: unforgeability under chosen message attacks and plaintext indistinguishability under chosen ciphertext attacks, against an insider adversary in a multi-user setting.

Most of the papers dealing with signcryption consider individual entities to perform the secret tasks of signcryption and unsigncryption. In many real-life situations, centralizing a secret task is not desirable due to both security and reliability reasons (a security / technical problem at a single entity can cause important threats / delays to the system). In these cases, a common approach is to decentralize the secret task(s) by considering a group of n entities, in such a way that the cooperation of at least t of them is necessary to successfully finish the task. This approach is known as (t, n)*-threshold cryptography.* In the scenario of signcryption, there are two secret tasks, so threshold cryptography could be applied to the signcryption protocol, to the unsigncryption protocol, or to both of them.

Among these three possibilities, here we focus on the situation where the unsigncryption task is distributed among a set of entities through a (t, n)-threshold process. Such schemes are known as *threshold unsigncryption* schemes. For simplicity we consider that the signcryption protocol is run by an individual entity (see however Section 9 for a discussion on fully threshold signcryption). We want to stress that the primitive of threshold unsigncryption is not just of theoretical interest; it has applications in real-life scenarios. For example, in a digital auction system, bidders may send their authenticated private bids, encrypted with the public key of a set of servers. In this way, even if some dishonest servers (less than t) collude, they will not be able to obtain information about the bids and influence the result of the auction. At the end of the auction, a large enough number of servers will cooperate to decrypt the bids and determine the winner of the auction and the price to pay.

The first works that focused on threshold unsigncryption [12, 26, 13, 14, 25] failed to achieve the desired security properties for this kind of schemes: existential unforgeability under chosen message attacks, and plaintext indistinguishability under chosen-ciphertext attacks (CCA), in a multi-user setting where the adversary can be insider and can corrupt up to $t - 1$ members of the target receiver entity. These security properties, along with the syntactic definition of threshold unsigncryption schemes, are detailed in Section 3. The security weaknesses of the above-mentioned threshold unsigncryption schemes were pointed out in [10, 20]. We showed in [10] (and we include this in Section 4 of this paper, for completeness) that even generic constructions of threshold unsigncryption schemes, obtained by combining a fully secure standard signature scheme and a fully secure threshold decryption scheme, do not achieve the maximum level of security. This is in contrast to what happens in the traditional scenario of signcryption, with a single receiver entity.

For this reason, one of the main goals in this area of threshold unsigncryption is to design new threshold unsigncryption schemes which are provably secure in the desired security model. The first two such schemes were proposed very recently: one scheme by ourselves in [10] which works in the traditional PKI setting, and one scheme in [20] which works in the identity-based setting. We include the design and security analysis of a slightly modified version of our scheme in [10], in Section 5 of this paper. Both our scheme and the scheme in [20] are proved secure in an idealized world, the *random oracle model*, where hash functions are assumed to behave as totally random functions. This assumption is useful but not achievable in real systems. Therefore, proofs in the random oracle model are just heuristic arguments, and thus security proofs in the standard model are preferable, when analyzing the security of cryptographic protocols.

To overcome this drawback, we propose and analyze in Section 6 a new threshold unsigncryption scheme; it is the first one in the literature which achieves, in the standard model, the required security properties. The design of this second scheme is quite modular: it employs two signature schemes and the ideas by Canetti-Halevi-Katz to achieve CCA security from identity-based selectively secure encryption [7].

The two schemes that we present in this paper, in Sections 5 and 6, have an additional property which may be of independent interest: the unsigncryption protocol of the schemes can be split into two parts. The first part, verifying the validity and the authorship of the ciphertext, can be done by anyone, because the required inputs are the ciphertext and the public key of the sender. The second part, decrypting the (valid) ciphertext, can be done without using the public key of the sender. To the best of our knowledge, these are the first fully secure signcryption schemes in the literature that enjoy this property, considering both individual and threshold (un)signcryption. This 'splitting' property seems to be very promising for applications requiring authentication and confidentiality, but also some level of anonymity or privacy in some of their phases. As an illustrative example, we explain in Section 8 the case of an electronic auction system.

Previous publication. We stress here that an earlier version of the results in Sections 4 and 5 was published in the Proceedings of the conference ProvSec'2010 [10]. The material in Sections 6, 7, 8 and 9 is completely original.

2 Preliminaries

In this section we recall some tools that will be used in the design and security analysis of the two threshold unsigncryption schemes that we present in this paper.

2.1 Strongly Unforgeable Signature Schemes

A signature scheme $\Theta = (\Theta.\mathsf{KG}, \Theta.\mathsf{Sign}, \Theta.\mathsf{Vfy})$ consists of three probabilistic polynomial time protocols. $\Theta\mathsf{KG}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$ is the key generation protocol, which takes as input a security parameter $\lambda \in \mathbb{N}$ and outputs a secret signing key sk and a public verification key vk. The signing protocol $\Theta.\mathsf{Sign}(\mathsf{sk}, m) \to \theta$ takes as input the signing key and a message m, and outputs a signature θ. Finally, the verification protocol $\Theta.\mathsf{Vfy}(\mathsf{vk}, m, \theta) \to 1$ or 0 takes as input the verification key, a message and a signature, and outputs 1 if the signature is valid, or 0 otherwise.

Regarding security, we consider an adversary F_Θ who first receives a verification key vk obtained from $\Theta.\mathsf{KG}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$. He can make at most q_S signature queries for messages m_i of his choice, obtaining as answer valid signatures $\Theta.\mathsf{Sign}(\mathsf{sk}, m_i) \to \theta_i$, and finally outputs a pair (m', θ'). We say that the adversary succeeds if $\Theta.\mathsf{Vfy}(\mathsf{vk}, m', \theta') \to 1$ and $(m', \theta') \neq (m_i, \theta_i)$ for all $i = 1, \ldots, q_S$.

We denote $\mathcal{F}_\Theta$'s success probability as $\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$. The signature scheme Θ is *strongly unforgeable* if $\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$ is a negligible function of the security parameter $\lambda \in \mathbb{N}$, for any polynomial-time attacker F_Θ against Θ. Here negligible means that $\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$ decreases (when λ increases, asymptotically) faster than the inverse of any polynomial. If a signature scheme is strongly unforgeable only against adversaries who can make at most $q_S = 1$ signature query, then the scheme is a secure *one-time* signature scheme.

An example of strongly unforgeable signature scheme can be found in [5]. The scheme therein is proved secure, in the standard model, under the Computational Diffie-Hellman Assumption (defined in the next subsection). Some examples of secure one-time signature schemes can be found in [17].

2.2 Bilinear Groups and Computational Assumptions

Given a security parameter $\lambda \in \mathbb{N}$, let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p, such that p is λ bits long.

The *Diffie-Hellman (DH, for short) problem* consists of computing the value g^{ab} from the values g, g^a, g^b, for random elements $a, b \in \mathbb{Z}_q^*$. The Diffie-Hellman Assumption states that the DH problem is hard to solve. A bit more formally, for any polynomial-time algorithm $\mathcal{A}^{DH}$ that receives as input $\mathbb{G}, g, g^a, g^b$, for random elements $a, b \in \mathbb{Z}_q^*$, we can define as $\mathsf{Adv}_{\mathcal{A}^{DH}}(\lambda)$ the probability that $\mathcal{A}^{DH}$ outputs the value g^{ab}. The *Diffie-Hellman Assumption* states that $\mathtt{Adv}_{\mathcal{A}^{DH}}(\lambda)$ is negligible in λ.

The Diffie-Hellman problem is easier to solve than the *Discrete Logarithm problem*: the input is $(\mathbb{G}, g, y)$, where $y \in \mathbb{G}$, and the goal for a solver $\mathcal{A}^{DL}$ is to find the integer $x \in \mathbb{Z}_q^*$ such that $y = g^x$. We can define $\mathtt{Adv}_{\mathcal{A}^{DL}}(\lambda)$ and the *Discrete Logarithm Assumption* analogously to the Diffie-Hellman case.

A group $\mathbb{G} = \langle g \rangle$ as defined above is said to be *bilinear* if there exist another group $\mathbb{G}_T$ with the same order p and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying the following properties:

1. $e(\cdot, \cdot)$ can be efficiently computed (in time polynomial in λ),
2. $e(g, g)$ is a generator of $\mathbb{G}_T$,
3. for any two elements $a, b \in \mathbb{Z}_p$, we have $e(g^a, g^b) = e(g, g)^{ab}$.

The *Decisional Bilinear Diffie-Hellman (DBDH, for short) problem* consists of distinguishing tuples of the form $(g, g^a, g^b, g^c, e(g, g)^{abc})$ from tuples of the form (g, g^a, g^b, g^c, T), for random $a, b, c \in \mathbb{Z}_p^*$ and random $T \in \mathbb{G}_T$. For any polynomial-time solver $\mathcal{A}^{DBDH}$ of this problem, we can define its advantage as $\mathtt{Adv}_{\mathcal{A}^{DBDH}}(\lambda) =$

$$\left| \Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 0] \; - \; \Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, T) = 0] \right|$$

The *Decisional Bilinear Diffie-Hellman Assumption* states that $\mathtt{Adv}_{\mathcal{A}^{DBDH}}(\lambda)$ is negligible in λ.

3 Signcryption with Threshold Unsigncryption

In a signcryption scheme, a user A sends a message to an intended receiver B, in a confidential and authenticated way: only B can obtain the original message, and B is convinced that the message comes from A. In a scenario where the role of B is distributed among a set of users, the cooperation of some authorized subset of users will be necessary to perform the unsigncryption phase. Each user in the set B will have a share of the secret information of B, and will use it to perform his part of the unsigncryption process. In this paper we will focus on threshold families of authorized subsets: the cooperation of at least t users in B will be necessary to successfully run the unsigncryption protocol. Both our formal definitions and our schemes can be extended to more general families of authorized subsets, by replacing threshold secret sharing techniques (i.e. Shamir's scheme [19]) with more general linear secret sharing schemes.

3.1 Syntactic Definition

A signcryption scheme with threshold unsigncryption $\Sigma = (\Sigma.\mathsf{St}, \Sigma.\mathsf{KG}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Uns})$ consists of four probabilistic polynomial-time algorithms:

- The randomized *setup* algorithm $\Sigma.\mathsf{St}$ takes a security parameter $\lambda \in \mathbb{N}$ and outputs some public parameters params that will be common to all the users in the system: the mathematical groups, generators, hash functions, etc. We write $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ to denote an execution of this algorithm.
- The *key generation* algorithm $\Sigma.\mathsf{KG}$ is different for an individual sender A than for a collective B of receivers. A single user A will get a pair $(\mathsf{sk}_A, \mathsf{pk}_A)$ of secret and public keys. In contrast, for a collective $B = \{B_1, \ldots, B_n\}$ of n users, the output will be a single public key pk_B for the group, and then a threshold secret share $\mathsf{sk}_{B,j}$ for each user B_j, for $j = 1, \ldots, n$, and for some threshold t such that $1 \leq t \leq n$. The key generation process for the collective B can be either run by a trusted third party, or by the users in B themselves. We will write $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, A, \text{'single'})$ and $(\{\mathsf{sk}_{B,j}\}_{1\leq j\leq n}, \mathsf{pk}_B) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, B, n, t, \text{'collective'})$ to refer to these two key generation protocols.
- The *signcryption* algorithm $\Sigma.\mathsf{Sign}$ takes as input params, a message M, the public key pk_B of the intended receiver group B, and the secret key sk_A of the sender. The output is a ciphertext C. We denote an execution of this algorithm as $C \leftarrow \Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_A)$.
- The *threshold unsigncryption* algorithm $\Sigma.\mathsf{Uns}$ is an interactive protocol run by some subset of users $B' \subset B$. The common inputs are params, a ciphertext C and the public key pk_A of the sender, whereas each user $B_j \in B'$ has as secret input his secret share $\mathsf{sk}_{B,j}$. The output is a message $\widetilde{M}$, which can eventually be the special symbol $\perp$, meaning that the ciphertext C is invalid. We write $\widetilde{M} \leftarrow \Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j\in B'})$ to refer to an execution of this protocol.

For correctness, condition $\Sigma.\mathsf{Uns}(\mathsf{params}, \Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_A), \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j\in B'}) = M$ is required, whenever B' contains at least t honest users and the values $\mathsf{params}, \mathsf{sk}_A, \mathsf{pk}_A, \{\mathsf{sk}_{B,j}\}_{1\leq j\leq n}, \mathsf{pk}_B$ have been obtained by properly executing the protocols $\Sigma.\mathsf{St}$ and $\Sigma.\mathsf{KG}$.

A different property that can be required is that of *robustness*, which informally means that dishonest receivers in B who do not follow the threshold unsigncryption protocol correctly can be detected and discarded, without affecting the correct completion of the protocol.

3.2 Security Model

A correct signcryption scheme must satisfy the security properties that are required for both encryption and signatures: confidentiality and unforgeability. In the threshold setting for unsigncryption, confidentiality must hold even if an attacker corrupts $t-1$ members of a collective of receivers. We consider a multi-user setting where an adversary is allowed to corrupt the maximum possible number of users (all except the target one), and where he can make both signcryption and unsigncryption queries for different users, messages and ciphertexts. In particular, unforgeability must hold even if the adversary knows the secret keys of all the possible collectives of receivers, and confidentiality must hold even if the adversary knows the secret keys of all the possible senders. In other words, we require *insider security*.

Note that we are considering only *static* adversaries, who choose the corrupted users at the beginning of the attack, in order to simplify the notation and thus allow a better understanding of the proposed schemes. In order to resist more powerful *adaptive* attacks, where the users may be corrupted at different stages of the system, our schemes should be combined with well-known techniques, as those in [6, 11, 16].

Unforgeability. *Unforgeability under chosen message attacks* is the standard security notion for signature schemes and in general for any cryptographic primitive which pretends to provide some kind of authentication or non-repudiation. The idea is that an attacker who does not know the secret key of a user A and who can ask A for some valid signatures (or, in our case, signcryptions) for messages of his choice must not be able to produce a different valid signature (signcryption) on behalf of A. For a security parameter $\lambda \in \mathbb{N}$, this notion is formalized by describing the following game that an attacker $\mathcal{A}_{\mathsf{UNF}}$ plays against a challenger:

1. The challenger runs $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ and gives params to $\mathcal{A}_{\mathsf{UNF}}$.
2. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target user $A^\star$. The challenger runs $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star}) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, A^\star, \text{'single'})$, keeps $\mathsf{sk}_{A^\star}$ private and gives $\mathsf{pk}_{A^\star}$ to $\mathcal{A}_{\mathsf{UNF}}$.
3. [**Queries**] $\mathcal{A}_{\mathsf{UNF}}$ can make adaptive queries to a signcryption oracle for sender $A^\star$: $\mathcal{A}_{\mathsf{UNF}}$ sends a tuple (M, pk_B) for some collective B of his choice, and obtains as answer $C \leftarrow \Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_{A^\star})$.
   Note that other kinds of queries (such as unsigncryption queries or signcryption queries for senders different from $A^\star$) make no sense because $\mathcal{A}_{\mathsf{UNF}}$ can reply such queries by himself.
4. [**Forgery**] Eventually, the attacker $\mathcal{A}_{\mathsf{UNF}}$ outputs a tuple $(\mathsf{pk}_{A^\star}, B^\star, \mathsf{pk}_{B^\star}, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star}, C^\star)$.

   We say that $\mathcal{A}_{\mathsf{UNF}}$ wins the game if:

– the protocol $\Sigma.\mathsf{Uns}(\mathsf{params}, C^\star, \mathsf{pk}_{A^\star}, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$ outputs a message $M^\star \neq \perp$,
– the tuple $(\mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star}, C^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ through a signcryption query.

The *advantage* of such an adversary $\mathcal{A}_{\mathsf{UNF}}$ in breaking the unforgeability of the signcryption scheme is defined as

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda) = \Pr[\mathcal{A}_{\mathsf{UNF}} \text{ wins}].$$

A signcryption scheme Σ with threshold unsigncryption is unforgeable if, for any polynomial time adversary $\mathcal{A}_{\mathsf{UNF}}$, the value $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$ is negligible with respect to the security parameter λ.

Indistinguishability. The confidentiality requirement for a signcryption scheme Σ with (t, n)-threshold unsigncryption (i.e. the fact that a signcryption on the message m addressed to B leaks no information on m to an attacker who only knows $t-1$ secret shares of sk_B) is ensured if the scheme enjoys the property of *indistinguishability under chosen ciphertext attacks* ($\mathsf{IND\text{-}CCA}$ security, for short). For a security parameter $\lambda \in \mathbb{N}$, this property is defined by considering the following game that an attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ plays against a challenger:

1. The challenger runs $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ and gives params to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.
2. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ chooses a target set $B^\star$ of n users and a subset $\widetilde{B} \subset B^\star$ of $t-1$ users, to be corrupted. The challenger runs $(\{\mathsf{sk}_{B^\star,j}\}_{1 \le j \le n}, \mathsf{pk}_{B^\star}) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, B^\star, n, t, \text{'collective'})$ and gives to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ the values $\mathsf{pk}_{B^\star}$ and $\{\mathsf{sk}_{B^\star,j}\}_{B_j \in \widetilde{B}}$. Without loss of generality, we can assume $B^\star = \{B_1, \dots, B_n\}$ and $\widetilde{B} = \{B_1, \dots, B_{t-1}\}$.
   Note that we are considering only *static* adversaries who choose the subset $\widetilde{B}$ of corrupted users at the beginning of the attack.
3. [**Queries**] $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can make adaptive queries to a threshold unsigncryption oracle for the target set $B^\star$: $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ sends a tuple (pk_A, C) for some public key pk_A of his choice. The challenger runs $\widetilde{M} \leftarrow \Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$. The attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ must be given all the information that is broadcast during the execution of this protocol $\Sigma.\mathsf{Uns}$, including $\widetilde{M}$.
   Other kinds of queries (such as unsigncryption queries for other collectives $B \neq B^\star$ or signcryption queries) make no sense because $\mathcal{A}_{\mathsf{UNF}}$ can reply such queries by himself.
4. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ chooses two messages M_0, M_1 of the same length, and a sender $A^\star$ along with $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star})$.
5. [**Challenge**] The challenger picks a random bit $d \in \{0, 1\}$, runs $C^\star \leftarrow \Sigma.\mathsf{Sign}(\mathsf{params}, M_d, \mathsf{pk}_{B^\star}, \mathsf{sk}_{A^\star})$ and gives $C^\star$ to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

6. Step 3 is repeated, with the restriction that the tuple $(\mathsf{pk}_{A^\star}, C^\star, B^\star)$ cannot be queried to the threshold unsigncryption oracle.
7. Finally, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs a bit d' as his guess of the bit d.

The advantage of such a (static) adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ in breaking the $\mathsf{IND\text{-}CCA}$ security of the signcryption scheme is defined as

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) = \left|\Pr[d' = d] - \frac{1}{2}\right|.$$

A signcryption scheme Σ with (t, n)-threshold unsigncryption is $\mathsf{IND\text{-}CCA}$ secure if $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$ is negligible with respect to the security parameter λ, for any polynomial time (static) adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

4 Generic Threshold Unsigncryption Schemes Are Not Fully Secure

The first proposals of explicit signcryption schemes with threshold unsigncryption that appeared in the literature did not achieve the full level of security described in the previous section. This includes two proposals [12, 26] in the traditional PKI setting, and three proposals [13, 14, 25] in the identity-based setting. Some details about the weaknesses of these schemes can be found in [10, 20].

It is a bit surprising that none of these first proposals considered the possibility of a generic construction of a signcryption scheme with threshold unsigncryption, following the well-known approaches Sign_then_Encrypt or Encrypt_then_Sign, that have been deeply analyzed in [1] for the case of individual signcryption. Therein, it is proved that both generic constructions achieve full security (against insider attackers in a multi-user setting) if the underlying signature and encryption schemes have full security. Thus, one could expect that the same happens in the scenario with threshold unsigncryption. But unfortunately this is not the case, as we argue below.

Let $\Omega = (\Omega.\mathsf{KG}, \Omega.\mathsf{Sign}, \Omega.\mathsf{Vfy})$ be a signature scheme, and $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{ThrDec})$ be a public encryption scheme with threshold decryption. For the keys of the generic signcryption schemes with threshold unsigncryption, an individual sender will run $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \Omega.\mathsf{KG}(1^\lambda)$ and a collective of receivers B will run $(\{\mathsf{sk}_{B,j}\}_{1\leq j\leq n}, \mathsf{pk}_B) \leftarrow \Pi.\mathsf{KG}(1^\lambda)$.

Let us consider for example the ThresholdEncrypt_then_Sign approach. To signcrypt a message m for the collective B, a sender A first computes $c \leftarrow \Pi.\mathsf{Enc}(\mathsf{pk}_B, m||\mathsf{pk}_A)$ and then signs $c||\mathsf{pk}_B$ to obtain $\omega \leftarrow \Omega.\mathsf{Sign}(\mathsf{sk}_A, c||\mathsf{pk}_B)$. The final ciphertext is $C = (c, \omega)$. To unsigncrypt such a ciphertext, members of B first verify the correctness of signature ω by running $\Omega.\mathsf{Vfy}(\mathsf{pk}_A, c||\mathsf{pk}_B, \omega)$. If the signature is not correct, the symbol $\perp$ is output. Otherwise, a subset $\tilde{B} \subset B$ of at least t members of B run $\Pi.\mathsf{ThrDec}(\{\mathsf{sk}_{B,j}\}_{B_j\in\tilde{B}}, c)$ to recover the message $m||\mathsf{pk}_A$. If the public key pk_A corresponds with that of the sender A, then m is the output of the protocol. If not, the output is $\perp$.

The $\mathsf{IND\text{-}CCA}$ security of this generic construction can be broken by an insider attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ in a multi-user scenario. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ receives a challenge ciphertext $C^\star = (c^\star, \omega^\star)$ for a challenge sender $A^\star$ and a challenge collective $B^\star$ of receivers. After that, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can generate keys $(\mathsf{sk}_A, \mathsf{pk}_A)$ for another user $A \neq A^\star$, compute a valid signature ω for $c^\star||\mathsf{pk}_{B^\star}$ using sk_A, and send $C = (c^\star, \omega)$ as a threshold unsigncryption query for sender A and collective $B^\star$ of receivers. As answer to this query, since the signature ω is valid, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ must receive all the information that the members of $B^\star$ would broadcast in the execution of the threshold decryption of $c^\star$. Even if the final output of this query is $\perp$, because the public key pk_A does not match the public key $\mathsf{pk}_{A^\star}$ which is encrypted in $c^\star$, the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ has obtained enough information to recover the whole plaintext encrypted in $c^\star$, and therefore succeeds in breaking the indistinguishability of the scheme. We stress that this same attack is valid against relaxed $\mathsf{IND\text{-}CCA}$ (see [8]), because the decryption of C (which is $\perp$) is different from the decryption of $C^\star$.

Regarding the Sign_then_ThresholdEncrypt approach, the attack is even simpler. Once $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ gets a challenge ciphertext $C^\star = c^\star$ for $A^\star$ and $B^\star$, where $c^\star$ is an encryption under Π of $(m, \omega^\star, \mathsf{pk}_{A^\star})$ and $\omega^\star$ is a signature on $m||\mathsf{pk}_{B^\star}$, all that $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ has to do is to make an unsigncryption query for the tuple $(C^\star, \mathsf{pk}_A, \mathsf{pk}_{B^\star})$, where $A \neq A^\star$. Even if the output of the protocol is again $\perp$, the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ gets all the partial information broadcast by the members of $B^\star$ in the execution of the threshold decryption of $c^\star$, which allows $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ to directly obtain the plaintext m.

5 A First New Threshold Unsigncryption Scheme with Full Security

This section is dedicated to the description and analysis of our first new signcryption scheme with (t,n)-threshold unsigncryption, achieving full security in the random oracle model. Our approach has been to take a secure public key encryption scheme with threshold decryption and modify it in order to accommodate also the authentication process. In particular, we have considered the scheme TDH1 of Shoup and Gennaro [22]. The idea of that scheme, to encrypt a message m for a collective B with public key pk_B, is to first compute a hashed ElGamal encryption (R,c) of m. That is, assuming that we have fixed a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, along with a hash function H_0, the sender computes $R = g^r$ and $c = m \oplus H_0((pk_B)^r)$. After that, he adds to the ciphertext another element $\bar{g} \in \mathbb{G}$ and the value $\bar{R} = \bar{g}^r$, and finally a zero-knowledge proof that $\mathsf{DiscLog}_g(R) = \mathsf{DiscLog}_{\bar{g}}(\bar{R})$. Members of B will start the real decryption process only if the proof of knowledge is valid.

Our signcryption scheme follows the same principle, but the sender A will compute now a zero-knowledge proof that both $\mathsf{DiscLog}_g(R) = \mathsf{DiscLog}_{\bar{g}}(\bar{R})$ holds and he knows sk_A such that $\mathsf{pk}_A = g^{\mathsf{sk}_A}$. We will prove that the resulting signcryption scheme (with threshold unsigncryption) enjoys the strong notions of unforgeability and indistinguishability. We consider for simplicity a scenario where the receivers follow the threshold unsigncryption protocol correctly. A simple modification of our scheme, by including appropriate non-interactive zero-knowledge proofs of the equality of two discrete logarithms, allows to provide robustness to the scheme against the action of malicious receivers. The protocols of the scheme are described below.

Setup: $\Sigma.\mathsf{St}(1^\lambda)$.
Given a security parameter λ, a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, such that q is λ bits long, is chosen. A length ℓ, which must be polynomial in λ, is defined for the maximum number of bits of the messages to be sent by the system. Three hash functions $H_0 : \{0,1\}^* \to \{0,1\}^\ell$, $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ are chosen. The output of the protocol is $\mathsf{params} = (q, \mathbb{G}, g, H_0, \ell, H_1, H_2)$.

Key Generation: $\Sigma.\mathsf{KG}(\mathsf{params}, A, \text{'single'})$ and $\Sigma.\mathsf{KG}(\mathsf{params}, B, n, t, \text{'collective'})$.
For an individual user A, the secret key sk_A is a random element in $\mathbb{Z}_q^*$, whereas the corresponding public key is $\mathsf{pk}_A = g^{\mathsf{sk}_A}$. The public output of this protocol is pk_A, and the secret output that is privately stored by A is sk_A.

For a collective $B = \{B_1, \ldots, B_n\}$ of n users, the common public key is computed as $\mathsf{pk}_B = g^{\mathsf{sk}_B}$ for some random value $\mathsf{sk}_B \in \mathbb{Z}_q^*$ that will remain unknown to the members of B. Each user $B_j \in B$ will receive a (t,n)-threshold share $\mathsf{sk}_{B,j}$ of sk_B, computed by using Shamir's secret sharing scheme [19]. This means that, for every subset $B' \subset B$ containing exactly t users, there exist values $\lambda_j^{B'} \in \mathbb{Z}_q^*$ such that $\mathsf{sk}_B = \sum_{B_j \in B'} \lambda_j^{B'} \mathsf{sk}_{B,j}$. The public output of this protocol is pk_B, whereas each user $B_j \in B$ receives a secret output $\mathsf{sk}_{B,j}$.

The key generation process for a collective B can be performed by a trusted dealer, or by the members of B themselves, by using some well-known techniques [9].

Both solutions permit that the values $D_{B,j} = g^{\mathsf{sk}_{B,j}}$ are made public, for $j = 1, \ldots, n$. These values would be necessary to provide robustness to the threshold unsigncryption process.

We assume that both pk_A and pk_B include descriptions of the identities of A and members of B.

Signcryption: $\Sigma.\mathsf{Sign}(\mathsf{params}, m, \mathsf{pk}_B, \mathsf{sk}_A)$.

1. Choose at random $r \in \mathbb{Z}_q^*$ and compute $R = g^r$.
2. Compute $k = H_0(R, \mathsf{pk}_B, (\mathsf{pk}_B)^r)$ and $c = m \oplus k$.
3. Choose at random $\alpha_1, \alpha_2 \in \mathbb{Z}_q^*$ and compute $Y_1 = g^{\alpha_1}$ and $Y_2 = g^{\alpha_2}$.
4. Compute $\bar{g} = H_1(c, R, Y_1, Y_2, \mathsf{pk}_A, \mathsf{pk}_B) \in \mathbb{G}$, and then $\bar{R} = \bar{g}^r$ and $\bar{Y}_1 = \bar{g}^{\alpha_1}$.
5. Compute $h = H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B)$.

6. Compute $s_1 = \alpha_1 - h \cdot r \bmod q$.
7. Compute $s_2 = \alpha_2 - h \cdot \mathsf{sk}_A \bmod q$.
8. Return the signcryption $C = (c, R, \bar{R}, h, s_1, s_2)$.

Threshold Unsigncryption: $\Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j \in B'})$.
Let $B' \subset B$ be a subset of users in B that want to cooperate to unsigncrypt a signcryption $C = (c, R, \bar{R}, h, s_1, s_2)$. They proceed as follows.

1. Each $B_j \in B'$ computes $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \mathsf{pk}_A, \mathsf{pk}_B)$.
2. Each $B_j \in B'$ checks if the following equality holds:
$$h \;=\; H_2(c, R, \bar{g}, \bar{R}, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \bar{g}^{s_1} \cdot \bar{R}^h, \mathsf{pk}_A, \mathsf{pk}_B)$$
3. If the equality does not hold, B_j broadcasts $(j, \perp)$.
4. Otherwise, $B_j \in B'$ broadcasts the value $T_j = R^{\mathsf{sk}_{B,j}}$.
If robustness was required, then B_j should also provide a non-interactive zero-knowledge proof that $\mathsf{DiscLog}_g(D_{B,j}) = \mathsf{DiscLog}_R(T_j)$.
5. If there are not t valid shares, then stop and output $\perp$. From t valid values T_j, different from $(j, \perp)$, recover the value R^{sk_B} by interpolation in the exponent: $R^{\mathsf{sk}_B} = \prod_{B_j \in B'} T_j^{\lambda_j^{B'}}$, where $\lambda_j^{B'} \in \mathbb{Z}_q$ are the Lagrange interpolation coefficients.
6. Compute $k = H_0(R, \mathsf{pk}_B, R^{\mathsf{sk}_B})$.
7. Return the value $m = c \oplus k$.

5.1 Security Analysis

Unforgeability. We are going to prove that our scheme enjoys unforgeability as long as the Discrete Logarithm problem is hard to solve. The proof is in the random oracle model for the hash function H_2.

Theorem 1. *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{\mathsf{UNF}}$ against the unforgeability of the new signcryption scheme, in the random oracle model, there exists a solver $\mathcal{A}^{DL}$ of the Discrete Logarithm problem such that*
$$\mathsf{Adv}_{\mathcal{A}^{DL}}(\lambda) \;\geq\; \mathcal{O}\left(\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)^2\right).$$

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{UNF}}$ that has advantage $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$ in breaking the unforgeability of our scheme, and assuming that the hash function H_2 behaves as a random oracle, we are going to construct an algorithm $\mathcal{A}^{DL}$ that solves the Discrete Logarithm problem in $\mathbb{G}$.

Let $(\mathbb{G}, y)$ be the instance of the Discrete Logarithm problem in $\mathbb{G} = \langle g \rangle$ that $\mathcal{A}^{DL}$ receives. The goal of $\mathcal{A}^{DL}$ is to find the integer $x \in \mathbb{Z}_q$ such that $y = g^x$. The algorithm $\mathcal{A}^{DL}$ initializes the attacker $\mathcal{A}_{\mathsf{UNF}}$ by giving $\mathsf{params} = (q, \mathbb{G}, g, H_0, \ell, H_1, H_2)$ to him. Here the hash functions $H_0 : \{0,1\}^* \to \{0,1\}^\ell$ and $H_1 : \{0,1\}^* \to \mathbb{G}$ are arbitrarily chosen by $\mathcal{A}^{DL}$. However, H_2 is modeled as a random oracle and so $\mathcal{A}^{DL}$ will maintain a table TAB_2 to answer the hash queries from $\mathcal{A}_{\mathsf{UNF}}$.

Key generation. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target sender $A^\star$ and requests the execution of the key generation protocol for this user. $\mathcal{A}^{DL}$ defines the public key of $A^\star$ as $\mathsf{pk}_{A^\star} = y$ and sends it to $\mathcal{A}_{\mathsf{UNF}}$. Note that the corresponding secret key $\mathsf{sk}_{A^\star}$, which is unknown to $\mathcal{A}^{DL}$, is precisely the solution to the given instance of the Discrete Logarithm problem.

Hash queries. Since H_2 is assumed to behave as a random function, $\mathcal{A}_{\mathsf{UNF}}$ can make queries $(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B)$ to the random oracle model for H_2. $\mathcal{A}^{DL}$ maintains a table TAB_2 to reply to these queries. TAB_2 contains two columns, one for the inputs and one for the corresponding outputs h of H_2. To reply the query $(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B)$, the algorithm $\mathcal{A}^{DL}$ checks if this input is already in TAB_2. If so, the matching output h is answered. If not, a random value $h \in \mathbb{Z}_q$ is chosen and answered to $\mathcal{A}_{\mathsf{UNF}}$, and the entry $H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B) = h$ is added to TAB_2.

Signcryption queries. $\mathcal{A}_{\mathsf{UNF}}$ can make signcryption queries for the sender $A^\star$, for pairs (m, pk_B) of his choice, where m is a message and B is a collective of receivers with public key pk_B. To reply to such queries, $\mathcal{A}^{DL}$ chooses at random a value $r \in \mathbb{Z}_q^*$ and computes $R = g^r$, $k = H_0(R, \mathsf{pk}_B, (\mathsf{pk}_B)^r)$ and $c = m \oplus k$. Then, $\mathcal{A}^{DL}$ must simulate a valid proof of knowledge to complete the rest of the ciphertext. To do this, $\mathcal{A}^{DL}$ acts as follows:

1. Choose at random $h, s_1, s_2 \in \mathbb{Z}_q$ and compute the values $Y_1 = g^{s_1} \cdot R^h$ and $Y_2 = g^{s_2} \cdot (\mathsf{pk}_{A^\star})^h$.
2. Compute $\bar{g} = H_1(c, R, Y_1, Y_2, \mathsf{pk}_{A^\star}, \mathsf{pk}_B)$, and then the values $\bar{R} = \bar{g}^r$ and $\bar{Y}_1 = \bar{g}^{s_1} \cdot \bar{R}^h$.
3. If the input $(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_{A^\star}, \mathsf{pk}_B)$ is already in TAB_2 (which happens with negligible probability), go back to Step 1.
4. Otherwise, 'falsely' add the relation $h = H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_{A^\star}, \mathsf{pk}_B)$ to TAB_2.

The final signcryption that $\mathcal{A}^{DL}$ sends to $\mathcal{A}_{\mathsf{UNF}}$ is $C = (c, R, \bar{R}, h, s_1, s_2)$.

Forgery. At some point, $\mathcal{A}_{\mathsf{UNF}}$ outputs a successful forgery; that is, a public key $\mathsf{pk}_{B^\star}$ and a signcryption $C^\star = (c^\star, R^\star, \bar{R}^\star, h^\star, s_1^\star, s_2^\star)$ such that:

– the protocol $\Sigma.\mathsf{Uns}(\mathsf{params}, C^\star, \mathsf{pk}_{A^\star}, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$ outputs $m^\star \neq \perp$,
– $(\mathsf{pk}_{A^\star}, m^\star, \mathsf{pk}_{B^\star}, C^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ during a signcryption query.

Since the forgery is valid, we must have $h^\star = H_2(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$, where $Y_1^\star = g^{s_1^\star} \cdot (R^\star)^{h^\star}$, $Y_2^\star = g^{s_2^\star} \cdot (\mathsf{pk}_{A^\star})^{h^\star}$ and $\bar{Y}_1^\star = (\bar{g}^\star)^{s_1^\star} \cdot (\bar{R}^\star)^{h^\star}$.

Furthermore, since the forgery is different from the ciphertexts obtained during the signcryption queries, we can be sure that the input $\mathsf{query}^\star = (c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ for H_2 has not been 'falsely' added by $\mathcal{A}^{DL}$ to TAB_2.

Replying the attack. Now the idea is to use the reply techniques introduced by Pointcheval and Stern in [18]. Without going into the details, $\mathcal{A}^{DL}$ will repeat the execution of the attacker $\mathcal{A}_{\mathsf{UNF}}$, with the same randomness but changing the values output by the random oracle H_2 from the query $\mathsf{query}^\star$ on.

With non-negligible probability (quadratic on the probability $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$ of the first successful forgery), the whole process run by $\mathcal{A}^{DL}$ would lead to two different successful forgeries $C^\star$ and $C'^\star$, for the same values of $c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star}$ (the input values for H_2), but with different H_2 outputs $h^\star \neq h'^\star$, and therefore (possibly different) values $s_1^\star, s_2^\star, s_1'^\star, s_2'^\star$.

We thus have

$$g^{s_2^\star} \cdot (\mathsf{pk}_{A^\star})^{h^\star} = Y_2^\star = g^{s_2'^\star} \cdot (\mathsf{pk}_{A^\star})^{h'^\star},$$

which leads to the relation $y = \mathsf{pk}_{A^\star} = \left(g^{s_2^\star - s_2'^\star}\right)^{1/(h'^\star - h^\star)}$.

Summing up, $\mathcal{A}^{DL}$ can output the value $x = \frac{s_2^\star - s_2'^\star}{h'^\star - h^\star} \bmod q$ as the solution to the given instance of the Discrete Logarithm problem. □

Indistinguishability. We reduce the $\mathsf{IND\text{-}CCA}$ security of the scheme to the hardness of solving the DH problem. The proof is in the random oracle model for the three hash functions H_0, H_1, H_2. The conclusion is that, under the Diffie Hellman Assumption for our group $\mathbb{G} = \langle g \rangle$, the new signcryption scheme has $\mathsf{IND\text{-}CCA}$ security.

Theorem 2. *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ against the $\mathsf{IND\text{-}CCA}$ security of the new signcryption scheme, in the random oracle model, there exists a solver $\mathcal{A}^{DH}$ of the Diffie-Hellman problem such that*

$$\mathsf{Adv}_{\mathcal{A}^{DH}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)/2.$$

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ that has advantage $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$ in breaking the $\mathsf{IND\text{-}CCA}$ security of our scheme, and assuming that hash functions H_0, H_1, H_2 behave as random oracles, we are going to construct an algorithm $\mathcal{A}^{DH}$ that solves the Diffie-Hellman problem.

$\mathcal{A}^{DH}$ receives as input $\mathbb{G}, g^a, g^b$, where $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order q. The goal of $\mathcal{A}^{DH}$ is to compute g^{ab}. $\mathcal{A}^{DH}$ initializes the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ by giving $\mathsf{params} = (q, \mathbb{G}, g, H_0, \ell, H_1, H_2)$ to him. Here the hash functions H_0, H_1 and H_2 will be modeled as random oracles; therefore, $\mathcal{A}^{DH}$ will maintain three tables TAB_0, TAB_1 and TAB_2 to answer the hash queries from $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Let $B^\star = \{B_1, \ldots, B_n\}$ be the target collective, and $\widetilde{B} = \{B_1, \ldots, B_{t-1}\} \subset B^\star$ be the subset of corrupted members of $B^\star$. The algorithm $\mathcal{A}^{DH}$ defines the public key of $B^\star$ as $\mathsf{pk}_{B^\star} = g^b$. This means that $\mathsf{sk}_{B^\star}$ is implicitly defined as b. For the corrupted members of $B^\star$, the shares $\{\mathsf{sk}_{B^\star,j}\}_{B_j \in \widetilde{B}}$ are chosen randomly and independently in $\mathbb{Z}_q$. Using interpolation in the exponent, all the values $D_{B^\star,j} = g^{\mathsf{sk}_{B^\star,j}}$ can be computed, for all the members $B_j \in B^\star$, corrupted or not.

Hash queries. $\mathcal{A}^{DH}$ creates and maintains three tables TAB_0, TAB_1 and TAB_2 to reply the hash queries from $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$. All the hash queries are processed by $\mathcal{A}^{DH}$ in the same way: given the input for a hash query, the algorithm $\mathcal{A}^{DH}$ checks if there already exists an entry in the corresponding table for that input. If this is the case, the existing output is answered. If this is not the case, a new output is chosen at random and answered to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$, and the new relation between input and output is added to the corresponding table.

For the particular case of H_1 queries, the corresponding outputs $\bar{g}$ are chosen as random powers of g^b. That is, $\mathcal{A}^{DH}$ chooses at random a fresh value $\beta \in \mathbb{Z}_q^*$ and computes the new output of H_1 as $\bar{g} = (g^b)^\beta$. The value β is stored as an additional value of the new entry in table TAB_1.

Whenever $\mathcal{A}^{DH}$ receives a H_0 query whose two first elements are g^a and g^b, the third element of the query is added to a different output table $\mathsf{TAB}^\star$, which will be the final output of $\mathcal{A}^{DH}$.

Unsigncryption queries. For an unsigncryption query (pk_A, C) sent for the target collective $B^\star$, where $C = (c, R, \bar{R}, h, s_1, s_2)$, the first thing to do is to check the validity of the zero-knowledge proof (h, s_1, s_2); that is, to check if $h = H_2(c, R, \bar{g}, \bar{R}, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \bar{g}^{s_1} \cdot \bar{R}^h, \mathsf{pk}_A, \mathsf{pk}_{B^\star})$, where $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \mathsf{pk}_A, \mathsf{pk}_{B^\star}) = (g^b)^\beta$, for some value β known by $\mathcal{A}^{DH}$. If this equation does not hold, then the answer to the query is $\perp$.

Otherwise, $\mathcal{A}^{DH}$ has to give to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ the values $R^{\mathsf{sk}_{B^\star,j}}$, for all $B_j \in B^\star$. For the corrupted members B_j, $j = 1, \ldots, t-1$, such values can be easily computed by $\mathcal{A}^{DH}$, because it knows $\mathsf{sk}_{B^\star,j}$. Note now that the value $R^{\mathsf{sk}_{B^\star}}$ can be computed by $\mathcal{A}^{DH}$ as $\bar{R}^{1/\beta}$. In effect, since the zero-knowledge proof is valid, this means that $\mathsf{DiscLog}_g(R) = \mathsf{DiscLog}_{\bar{g}}(\bar{R})$, where $\bar{g} = g^{b\beta}$, and so $R^{b\beta} = \bar{R}$. Now, knowing $R^{\mathsf{sk}_{B^\star}}$ and $R^{\mathsf{sk}_{B^\star,j}}$ for $j = 1, \ldots, t-1$, the algorithm $\mathcal{A}^{DH}$ can compute the rest of values $R^{\mathsf{sk}_{B^\star,j}}$, for $j = t, t+1, \ldots, n$, by interpolation in the exponent. Once this is done, the rest of the unsigncryption process can be easily completed by $\mathcal{A}^{DH}$, who obtains a message m and sends all this information to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Challenge. At some point, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs two messages m_0, m_1 of the same length, along with a key pair $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star})$ for a sender $A^\star$. To produce the challenge ciphertext $C^\star$, the algorithm $\mathcal{A}^{DH}$ defines $R^\star = g^a$ and then chooses at random the values $c^\star \in \{0,1\}^\ell$, $h^\star, s_1^\star, s_2^\star \in \mathbb{Z}_q$ and $\beta^\star \in \mathbb{Z}_q^*$. After that, $\mathcal{A}^{DH}$ defines $\bar{g}^\star = g^{\beta^\star}$, $\bar{R}^\star = (g^a)^{\beta^\star}$, $Y_1^\star = g^{s_1^\star} \cdot (R^\star)^{h^\star}$, $Y_2^\star = g^{s_2^\star} \cdot (\mathsf{pk}_{A^\star})^{h^\star}$ and $\bar{Y}_1^\star = \bar{g}^{s_1^\star} \cdot (\bar{R}^\star)^{h^\star}$.

If either the input $(c^\star, R^\star, Y_1^\star, Y_2^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ already exists in TAB_1, or the input $(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ already exists in TAB_2, the algorithm $\mathcal{A}^{DH}$ goes back to choose at random other values for $c^\star, h^\star$, etc. Finally, the relation $\bar{g}^\star = H_1(c^\star, R^\star, Y_1^\star, Y_2^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ is added to TAB_1 and the relation $h^\star = H_2(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ is added to TAB_2. The challenge ciphertext that $\mathcal{A}^{DH}$ sends to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ is $C^\star = (c^\star, R^\star, \bar{R}^\star, h^\star, s_1^\star, s_2^\star)$.

More unsigncryption queries. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can make more hash and unsigncryption queries, which are answered exactly in the same way as described before the challenge phase. The only delicate point is that $\mathcal{A}^{DH}$ could not answer to a valid unsigncryption query $C = (c, R, \bar{R}, h, s_1, s_2)$ for which the value of $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \mathsf{pk}_A, \mathsf{pk}_{B^\star}) = \bar{g}^\star$, because this value does not have the necessary form $(g^b)^\beta$. But this happens only if the two inputs of H_1, in both the challenge ciphertext and in this queried ciphertext, are the same. Since both zero-knowledge proofs are valid, we would also have that the value of $\bar{R}$ is equal in both cases, and therefore the values of $h, s_1, s_2, \mathsf{pk}_A$ would also be equal. The conclusion is that the unsigncryption query C would be exactly the challenge ciphertext, and this query is prohibited to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Final analysis. Finally, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs a guess bit b'. We are assuming that $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ succeeds with probability significantly greater than $1/2$ (random guess). Since H_0 is assumed to behave as a random function, this can happen only if $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ has asked to the random oracle H_0 the input corresponding to the challenge $C^\star$. This input is (g^a, g^b, g^{ab}). Therefore, with non-negligible probability $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)/2$, the value g^{ab} is in the table $\mathsf{TAB}^\star$ constructed by $\mathcal{A}^{DH}$, and therefore the output of $\mathcal{A}^{DH}$ contains the correct answer for the given instance of the DH problem. As the authors of [22] indicate, we could use the Diffie-Hellman self-corrector described in [21] to transform this algorithm $\mathcal{A}^{DH}$ into an algorithm that only outputs the single and correct solution to the DH problem. □

6 A Threshold Unsigncryption Scheme with Full Security in the Standard Model

The security of the scheme in the previous section has been proved in the random oracle model, which is an heuristic model, not a realistic one. Therefore, schemes enjoying security in the standard model are much preferable. We design and analyze in this section the first signcryption scheme with (t, n)-threshold unsigncryption enjoying full security in the standard model.

The rationale for the design of this second scheme is the following one. Boneh, Boyen and Halevi showed in [4] how to design threshold decryption schemes with CCA security in the standard model, by adapting the strategy proposed by Canetti, Halevi and Katz in [7]. That is, to encrypt a message M, a key-pair $(\tilde{\mathsf{sk}}, \tilde{\mathsf{vk}})$ for some strongly secure one-time signature scheme is generated, then $\tilde{\mathsf{vk}}$ is used to derive an identity id, and message M is encrypted for identity id, by using a selectively-secure identity-based encryption scheme such as that in [3]. The resulting ciphertext $\tilde{C}$ is signed with $\tilde{\mathsf{sk}}$, leading to a signature $\tilde{\theta}$. Both $\tilde{\mathsf{vk}}$ and $\tilde{\theta}$ are added to $\tilde{C}$. The set of receivers share the master secret key of the identity-based encryption scheme. To decrypt, they first verify that the signature $\tilde{\theta}$ on $\tilde{C}$ is correct under key $\tilde{\mathsf{vk}}$; if this is the case, they can cooperate to derive the secret key for identity id and then decrypt $\tilde{C}$ to recover the plaintext M.

To implement the primitive of signcryption with threshold unsigncryption, our idea is that the sender A signs the message $\tilde{C}||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}}$ with a strongly secure signature scheme, obtaining θ, and then the (one-time) signature $\tilde{\theta}$ is computed on $\tilde{C}||\mathsf{pk}_A||\mathsf{pk}_B||\theta$. The final signcryption is $C = (\tilde{C}, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$. With this technique, the receivers will be convinced of the authorship of sender A because even insider attacks can be prevented.

Although we could have described a more generic construction by using in a black-box way the primitives of (one-time) signature schemes and identity-based encryption with threshold key generation, it turns out that the only realization of the later primitive in the standard model is the specific scheme in [4], using bilinear pairings. For this reason, and for the sake of clarity in the presentation, we have decided to describe the new signcryption scheme directly instantiated with the pairing-based scheme in [4]. The protocols of the scheme are detailed below.

Setup: $\Sigma.\mathsf{St}(1^\lambda)$.
Given $\lambda \in \mathbb{N}$, a cyclic bilinear group $\mathbb{G} = \langle g \rangle$ of prime order p, such that p is λ bits long, is chosen. This means that there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ for some group $\mathbb{G}_T$. Let $H : \{0,1\}^* \to \mathbb{Z}_p^*$ be a collision-resistant hash function. Two more generators $h, g_2 \in \mathbb{G}$ are randomly selected.

Let $\Theta = (\Theta.\mathsf{KG}, \Theta.\mathsf{Sign}, \Theta.\mathsf{Vfy})$ be a strongly unforgeable signature scheme, and let $\tilde{\Theta} = (\tilde{\Theta}.\mathsf{KG}, \tilde{\Theta}.\mathsf{Sign}, \tilde{\Theta}.\mathsf{Vfy})$ be a strongly secure one-time signature scheme. Note that we could take $\tilde{\Theta} = \Theta$.

The output of the protocol is $\mathsf{params} = (p, \mathbb{G}, g, \mathbb{G}_T, e, H, h, g_2, \Theta, \tilde{\Theta})$.

Key Generation: $\Sigma.\mathsf{KG}(\mathsf{params}, A, \text{`single'})$ and $\Sigma.\mathsf{KG}(\mathsf{params}, B, n, t, \text{`collective'})$.
For an individual user A, the key generation protocol of the signature scheme Θ is executed, and the resulting signing and verification keys are defined as the secret and public keys for user A. That is, $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \Theta.\mathsf{KG}(1^\lambda)$.

For a collective $B = \{B_1, \ldots, B_n\}$ of n users, the common public key is computed as $\mathsf{pk}_B = g^{\alpha_B}$ for some random value $\alpha_B \in \mathbb{Z}_p^*$ that will remain unknown to the members of B. This value α_B is distributed in shares $\{\alpha_{B,j}\}_{B_j \in B}$ through Shamir's (t, n)-threshold secret sharing scheme [19]. In particular, for every subset $B' \subset B$ containing at least t users, there exist values $\lambda_j^{B'} \in \mathbb{Z}_q^*$ such that $\alpha_B = \sum\limits_{B_j \in B'} \lambda_j^{B'} \alpha_{B,j}$. The public output of this protocol is pk_B, whereas each user $B_j \in B$ privately receives and stores his share $\mathsf{sk}_{B,j} = g_2^{\alpha_{B,j}}$ of the secret key $\mathsf{sk}_B = g_2^{\alpha_B}$. Again, the key generation process for a collective B can be performed by a trusted dealer, or by the members of B themselves, by using the techniques in [9].

Both solutions allow the publication of the values $D_{B,j} = g^{\alpha_{B,j}}$, for $j = 1, \ldots, n$. These values would be necessary if robustness of the threshold unsigncryption process was required.

Signcryption: $\Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_A)$, where $M \in \mathbb{G}_T$.

1. Run $(\tilde{\mathsf{sk}}, \tilde{\mathsf{vk}}) \leftarrow \tilde{\Theta}.\mathsf{KG}(1^\lambda)$ to obtain an ephemeral pair of signing and verification keys for the one-time signature scheme $\tilde{\Theta}$.
2. Derive $\mathsf{id} = H(\tilde{\mathsf{vk}})$, which is an element in $\mathbb{Z}_p^*$.
3. Choose at random $s \in \mathbb{Z}_p^*$.
4. Compute $C_1 = g^s$, $C_2 = M \cdot e(\mathsf{pk}_B, g_2)^s$ and $C_3 = \left(\mathsf{pk}_B^{\mathsf{id}} \cdot h\right)^s$.
5. Use sk_A to compute a signature θ on the message $C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}}$ for the scheme Θ. That is, $\theta \leftarrow \Theta.\mathsf{Sign}(\mathsf{sk}_A,\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}})$.
6. Use the ephemeral secret key $\tilde{\mathsf{sk}}$ to compute a signature $\tilde{\theta}$ on the message $C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta$ for the scheme $\tilde{\Theta}$. That is, $\tilde{\theta} \leftarrow \tilde{\Theta}.\mathsf{Sign}(\tilde{\mathsf{sk}},\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta)$.
7. Return the signcryption $C = (C_1, C_2, C_3, \mathsf{vk}, \theta, \tilde{\theta})$.

Threshold Unsigncryption: $\Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j \in B'})$.
Let $B' \subset B$ be a subset of users in B that want to cooperate to unsigncrypt a signcryption $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$ sent by user A. They proceed as follows.

1. Each $B_j \in B'$ runs $\tilde{\Theta}.\mathsf{Vfy}(\tilde{\mathsf{vk}},\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta\ , \tilde{\theta})$. If the output is 0, he broadcasts $(j, \perp)$.
2. Each $B_j \in B'$ runs $\Theta.\mathsf{Vfy}(\mathsf{pk}_A,\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}}\ , \theta)$. If the output is 0, he broadcasts $(j, \perp)$.
3. Each $B_j \in B'$ derives $\mathsf{id} = H(\tilde{\mathsf{vk}})$ and checks if $e(C_3, g) = e(\mathsf{pk}_B^{\mathsf{id}} \cdot h, C_1)$. If this equality does not hold, B_j broadcasts $(j, \perp)$.
4. Each $B_j \in B'$ chooses $r_j \in \mathbb{Z}_p$ at random and broadcasts the tuple $(j, \omega_{0,j}, \omega_{1,j})$, where
$$\omega_{0,j} = \mathsf{sk}_{B,j} \cdot (\mathsf{pk}_B^{\mathsf{id}} \cdot h)^{r_j} \quad \text{and} \quad \omega_{1,j} = g^{r_j}$$
If robustness was required, the correctness of this tuple could be publicly verified by checking if $e(\omega_{0,j}, g) = e(D_{B,j}, g_2) \cdot e(\mathsf{pk}_B^{\mathsf{id}} \cdot h, \omega_{1,j})$.
5. If there are not t valid shares, then stop and output $\perp$. From t valid tuples $\{(j, \omega_{0,j}, \omega_{1,j})\}_{B_j \in B'}$, one can consider the Lagrange interpolation coefficients $\lambda_j^{B'} \in \mathbb{Z}_q$ such that $\mathsf{sk}_B = \prod\limits_{B_j \in B'} \mathsf{sk}_{B,j}^{\lambda_j^{B'}}$.
6. Compute $\omega_0 = \prod\limits_{B_j \in B'} \omega_{0,j}^{\lambda_j^{B'}}$ and $\omega_1 = \prod\limits_{B_j \in B'} \omega_{1,j}^{\lambda_j^{B'}}$.
[Note that $\omega_0 = \mathsf{sk}_B \cdot (\mathsf{pk}_B^{\mathsf{id}} \cdot h)^{\tilde{r}}$ and $\omega_1 = g^{\tilde{r}}$, being $\tilde{r} = \sum\limits_{B_j \in B'} \lambda_j^{B'} r_j$.]
7. Return the message $M = C_2 \cdot \frac{e(C_3, \omega_1)}{e(C_1, \omega_0)}$.

It is important to point out that the threshold unsigncryption protocol is non-interactive, in the sense that each receiver B_j can do his secret part of the unsigncryption task independently of the other receivers.

6.1 Security Analysis

Unforgeability. We are going to prove that the scheme enjoys unforgeability as long as the signature schemes Θ and $\tilde{\Theta}$ are strongly unforgeable.

Theorem 3. *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{\mathsf{UNF}}$ against the unforgeability of the new signcryption scheme, making Q signcryption queries, there exists an attacker $\mathcal{F}_\Theta$ against Θ or an attacker $\mathcal{F}_{\tilde{\Theta}}$ against $\tilde{\Theta}$, such that $\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda) + Q \cdot \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$.*

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{UNF}}$ against the unforgeability of the scheme, we are going to construct a forger $\mathcal{F}_\Theta$ against the signature scheme Θ.

$\mathcal{F}_\Theta$ receives as input a verification key vk obtained from an execution $(\mathsf{sk}, \mathsf{vk}) \leftarrow \Theta.\mathsf{KG}(1^\lambda)$, and has access to a signing oracle $\Theta.\mathsf{Sign}(\mathsf{sk}, \cdot)$ for messages of its choice. The algorithm $\mathcal{F}_\Theta$ runs the setup protocol $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ and initializes the attacker $\mathcal{A}_{\mathsf{UNF}}$ by giving params to it.

Key generation. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target sender $A^\star$ and requests the execution of the key generation protocol for this user. $\mathcal{F}_\Theta$ defines the public key of $A^\star$ as $\mathsf{pk}_{A^\star} = \mathsf{vk}$ and sends it to $\mathcal{A}_{\mathsf{UNF}}$.

Signcryption queries. $\mathcal{A}_{\mathsf{UNF}}$ can make signcryption queries for the sender $A^\star$, for pairs (M_i, pk_{B_i}) of its choice, where M_i is a message and B_i is a collective of receivers with public key pk_{B_i}. To reply such queries, $\mathcal{F}_\Theta$ runs steps 1-4 of the signcryption protocol $\Sigma.\mathsf{Sign}(\mathsf{params}, M_i, \mathsf{pk}_{B_i}, \mathsf{sk}_{A^\star})$, obtaining consistent values $\tilde{\mathsf{sk}}_i, \tilde{\mathsf{vk}}_i, C_{1,i}, C_{2,i}, C_{3,i}$. After that, $\mathcal{F}_\Theta$ queries its signing oracle with message $m_i = C_{1,i}||C_{2,i}||C_{3,i}||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B_i}||\tilde{\mathsf{vk}}_i$, and obtains as answer a valid signature θ_i for the signature scheme Θ and public key $\mathsf{pk}_{A^\star}$.

Then, $\mathcal{F}_\Theta$ can run step 6 of the signcryption protocol: $\tilde{\theta}_i \leftarrow \tilde{\Theta}.\mathsf{Sign}(\tilde{\mathsf{sk}}_i,\ C_{1,i}||C_{2,i}||C_{3,i}||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B_i}||\theta_i)$. The final signcryption that $\mathcal{F}_\Theta$ sends to $\mathcal{A}_{\mathsf{UNF}}$ is $C_i = (C_{1,i}, C_{2,i}, C_{3,i}, \tilde{\mathsf{vk}}_i, \theta_i, \tilde{\theta}_i)$.

Forgery. At some point, and with probability $\varepsilon = \mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$, the attacker $\mathcal{A}_{\mathsf{UNF}}$ outputs a successful forgery; that is, a public key $\mathsf{pk}_{B^\star}$ and a signcryption $C^\star = (C_1^\star, C_2^\star, C_3^\star, \tilde{\mathsf{vk}}^\star, \theta^\star, \tilde{\theta}^\star)$ such that:

– the protocol $\Sigma.\mathsf{Uns}(\mathsf{params}, C^\star, \mathsf{pk}_{A^\star}, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$ outputs $M^\star \neq \perp$,
– $(\mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star}, C^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ during a signcryption query.

Let us define $m^\star = C_1^\star||C_2^\star||C_3^\star||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B^\star}||\tilde{\mathsf{vk}}^\star$. We can distinguish two cases.

First, with probability ε_1 we can have $(m^\star, \theta^\star) \neq (m_i, \theta_i)$, for all messages m_i that $\mathcal{F}_\Theta$ has queried to its signing oracle. Then $\mathcal{F}_\Theta$ has obtained a valid and new signature $(m^\star, \theta^\star)$ for the scheme Θ and public key $\mathsf{pk}_{A^\star}$. Therefore, $\varepsilon_1 \leq \mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$.

Otherwise, with probability $\varepsilon_2 = \varepsilon - \varepsilon_1$, we can have $(m^\star, \theta^\star) = (m_i, \theta_i)$ for some of the Q messages m_i that $\mathcal{F}_\Theta$ has queried to its signing oracle. In this case, since the forgery by $\mathcal{A}_{\mathsf{UNF}}$ is valid, the only possibility is $\tilde{\theta}^\star \neq \tilde{\theta}_i$. In this case, it is easy to construct a forger $\mathcal{F}_{\tilde{\Theta}}$ against the strong one-time unforgeability of $\tilde{\Theta}$: this forger receives as input a target verification key $\tilde{\mathsf{vk}}'$, then guesses the correct signcryption query i, uses its only access to a signing oracle to obtain the corresponding signature $\tilde{\theta}_i$ for this query, and uses other ephemeral key pairs $(\tilde{\mathsf{sk}}, \tilde{\mathsf{vk}})$ to reply the rest of signcryption queries. If the guess of i is correct (which happens with probability $1/Q$), then this second kind of forgery by $\mathcal{A}_{\mathsf{UNF}}$ leads to a valid forgery by $\mathcal{F}_{\tilde{\Theta}}$ against scheme $\tilde{\Theta}$. Therefore, we have $\mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda) \geq \varepsilon_2/Q$.

Summing up, we have $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda) = \varepsilon = \varepsilon_1 + \varepsilon_2 \leq \mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda) + Q \cdot \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda)$, as desired. □

Indistinguishability. We reduce the $\mathsf{IND\text{-}CCA}$ security of the scheme to the hardness of solving the DBDH problem in groups $\mathbb{G}, \mathbb{G}_T$ and to the security of the underlying signature scheme $\tilde{\Theta}$, which we assume to be one-time strongly secure. The proof is in the standard model.

Theorem 4. *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ against the IND-CCA security of the new signcryption scheme, there exists a solver $\mathcal{A}^{DBDH}$ of the Decisional Bilinear Diffie-Hellman problem or an attacker $F_{\tilde{\Theta}}$ against $\tilde{\Theta}$ such that $\mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$.*

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ that has advantage $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$ in breaking the IND-CCA security of our scheme, we construct an algorithm $\mathcal{A}^{DBDH}$ that solves the Decisional Bilinear Diffie-Hellman problem in groups $\mathbb{G}, \mathbb{G}_T$.

$\mathcal{A}^{DBDH}$ receives as input g^a, g^b, g^c, R, where R is either $e(g,g)^{abc}$ or a random element in $\mathbb{G}_T$. The goal of $\mathcal{A}^{DBDH}$ is to distinguish between these two cases.

$\mathcal{A}^{DBDH}$ runs the key generation protocol for the signature scheme $\tilde{\Theta}$, obtaining $(\tilde{\mathsf{sk}}^\star, \tilde{\mathsf{vk}}^\star) \leftarrow \tilde{\Theta}.\mathsf{KG}(1^\lambda)$. Then $\mathcal{A}^{DBDH}$ chooses at random a suitable hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$ and a suitable signature scheme Θ. The value $\mathsf{id}^\star = H(\tilde{\mathsf{vk}}^\star)$ is computed. $\mathcal{A}^{DBDH}$ defines $g_2 = g^a$, chooses at random $\gamma \in \mathbb{Z}_p^*$ and defines $h = (g^b)^{-\mathsf{id}^\star} \cdot g^\gamma$. Then $\mathcal{A}^{DBDH}$ initializes the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ by giving $\mathsf{params} = (p, \mathbb{G}, g, \mathbb{G}_T, e, H, h, g_2, \Theta, \tilde{\Theta})$ to it.

Key generation. Let $B^\star = \{B_1, \ldots, B_n\}$ be the target collective and $\widetilde{B} = \{B_1, \ldots, B_{t-1}\} \subset B^\star$ be the subset of corrupted members of $B^\star$, chosen by $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$. The algorithm $\mathcal{A}^{DBDH}$ defines the public key of $B^\star$ as $\mathsf{pk}_{B^\star} = g^b$. This means that the secret value $\alpha_{B^\star}$ is implicitly defined as b. For the corrupted members of $B^\star$, the shares $\{\mathsf{sk}_{B^\star,j}\}_{B_j \in \widetilde{B}}$ are computed by first choosing random and independent values $\alpha_{B^\star,j} \in \mathbb{Z}_p$ and then computing $\mathsf{sk}_{B^\star,j} = g_2^{\alpha_{B^\star,j}}$. Let $f \in \mathbb{Z}_p[X]$ be the implicit polynomial, with degree $t-1$, that satisfies $f(0) = b = \alpha_{B^\star}$ and $f(j) = \alpha_{B^\star,j}$ for $j = 1, \ldots, t-1$.

Using interpolation in the exponent and the values $\mathsf{pk}_{B^\star} = g^{\alpha_{B^\star}} = g^b$ and $\{\alpha_{B^\star,j}\}_{B_j \in \widetilde{B}}$, all the values $D_{B^\star,j} = g^{\alpha_{B^\star,j}}$ could be obtained (if robustness was required) for all the members $B_j \in B^\star$.

Unsigncryption queries. Let (pk_A, C) be an unsigncryption query sent for the target collective $B^\star$, where $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$. If $\tilde{\mathsf{vk}} = \tilde{\mathsf{vk}}^\star$ and $1 \leftarrow \tilde{\Theta}.\mathsf{Vfy}(\tilde{\mathsf{vk}},\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta\ , \tilde{\theta})$, then $\mathcal{A}^{DBDH}$ aborts and outputs a random bit. Otherwise, $\mathcal{A}^{DBDH}$ runs steps 1-3 (which are public verifications) of the unsigncryption protocol.

If (pk_A, C) is a valid signcryption and $\mathcal{A}^{DBDH}$ has not aborted, we have $\tilde{\mathsf{vk}} \neq \tilde{\mathsf{vk}}^\star$ and $\mathsf{id} = H(\tilde{\mathsf{vk}})$. Since the hash function is assumed to be collision-resistant, we have $\mathsf{id} \neq \mathsf{id}^\star$ as well. Now $\mathcal{A}^{DBDH}$ is required to simulate the values that would be broadcast in an execution of the rest of the protocol. This means simulating consistent tuples $(j, \omega_{0,j}, \omega_{1,j})$ for any $B_j \in B^\star$, where

$$\omega_{0,j} = \mathsf{sk}_{B^\star,j} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{r_j} \quad \text{and} \quad \omega_{1,j} = g^{r_j}$$

for some (randomly uniform) value $r_j \in \mathbb{Z}_p$. For the corrupted members B_j, $j = 1, \ldots, t-1$, such values can be easily computed by $\mathcal{A}^{DBDH}$, because it knows $\mathsf{sk}_{B^\star,j}$.

For any non-corrupted member B_i, $i = t, \ldots, n$, let $\lambda_0, \lambda_1, \ldots, \lambda_{t-1} \in \mathbb{Z}_p$ be the Lagrange interpolation coefficients corresponding to the set $\{0, 1, \ldots, t-1\}$ and interpolation point i. These coefficients can be publicly computed because they are independent of the (unknown) polynomial f. We have $f(i) = \lambda_0 f(0) + \sum_{j=1}^{t-1} \lambda_j f(j)$. Now $\mathcal{A}^{DBDH}$ can choose a random $\tilde{r}_i \in \mathbb{Z}_p$ and define

$$\omega_{0,i} = g_2^{\frac{-\gamma\lambda_0}{\mathsf{id}-\mathsf{id}^\star}} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{\tilde{r}_i} \cdot \prod_{j=1}^{t-1} \mathsf{sk}_{B^\star,j}^{\lambda_j} \quad \text{and} \quad \omega_{1,i} = g_2^{\frac{-\lambda_0}{\mathsf{id}-\mathsf{id}^\star}} \cdot g^{\tilde{r}_i}$$

It is not difficult to see that these two values $(\omega_{0,i}, \omega_{1,i})$ have the form

$$\omega_{0,i} = g_2^{f(i)} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{r_i} = \mathsf{sk}_{B^\star,i} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{r_i} \quad \text{and} \quad \omega_{1,i} = g^{r_i},$$

being $r_i = \tilde{r}_i - \frac{a\lambda_0}{\mathsf{id}-\mathsf{id}^\star}$ an implicit but randomly uniform value in $\mathbb{Z}_p$.

Summing up, $\mathcal{A}^{DBDH}$ can simulate valid tuples $(j, \omega_{0,j}, \omega_{1,j})$ for any $B_j \in B^\star$. Once this is done, the rest of the unsigncryption process can be easily completed by $\mathcal{A}^{DBDH}$, who obtains a message M and sends all this information to $\mathcal{A}_{\textsf{IND-CCA}}$.

Challenge. At some point, $\mathcal{A}_{\textsf{IND-CCA}}$ outputs two messages M_0, M_1 of the same length, along with a key pair $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star})$ for a sender $A^\star$. To produce the challenge ciphertext $C^\star$, the algorithm $\mathcal{A}^{DBDH}$ chooses at random a bit $d \in \{0,1\}$, and proceeds as follows.

1. Define $C_1^\star = g^c$, $C_2^\star = M_d \cdot R$ and $C_3^\star = (g^c)^\gamma = \ldots = (\mathsf{pk}_{B^\star}^{\mathsf{id}^\star} \cdot h)^c$.
   Note that $(C_1^\star, C_2^\star, C_3^\star)$ is a consistent encryption of M_b for identity $\mathsf{id}^\star$ when $R = e(g,g)^{abc}$. On the other hand, when $R \in \mathbb{G}_T$ is random, the distribution of $(C_1^\star, C_2^\star, C_3^\star)$ is independent of the bit d.
2. Run $\theta^\star \leftarrow \Theta.\mathsf{Sign}(\mathsf{sk}_{A^\star},\ C_1^\star || C_2^\star || C_3^\star || \mathsf{pk}_{A^\star} || \mathsf{pk}_{B^\star} || \tilde{\mathsf{vk}}^\star)$.
3. Run $\tilde{\theta}^\star \leftarrow \tilde{\Theta}.\mathsf{Sign}(\tilde{\mathsf{sk}}^\star,\ C_1^\star || C_2^\star || C_3^\star || \mathsf{pk}_{A^\star} || \mathsf{pk}_{B^\star} || \theta^\star)$.
4. Send to $\mathcal{A}_{\textsf{IND-CCA}}$ the challenge signcryption $C^\star = (C_1^\star, C_2^\star, C_3^\star, \tilde{\mathsf{vk}}^\star, \theta^\star, \tilde{\theta}^\star)$.

More unsigncryption queries. $\mathcal{A}_{\textsf{IND-CCA}}$ can make more unsigncryption queries $(\mathsf{pk}_A, C) \neq (\mathsf{pk}_{A^\star}, C^\star)$ for the target collective $B^\star$, where $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$, as long as the challenge signcryption is not queried. If $\tilde{\mathsf{vk}} \neq \tilde{\mathsf{vk}}^\star$, then these queries are handled in the same way as explained above.

Otherwise, if $\tilde{\mathsf{vk}} = \tilde{\mathsf{vk}}^\star$ and $1 \leftarrow \tilde{\Theta}.\mathsf{Vfy}(\tilde{\mathsf{vk}},\ C_1 || C_2 || C_3 || \mathsf{pk}_A || \mathsf{pk}_B || \theta\ , \tilde{\theta})$, then $\mathcal{A}^{DBDH}$ aborts and outputs a random bit.

Final analysis. Finally, $\mathcal{A}_{\textsf{IND-CCA}}$ outputs a guess bit d'. If $d' = d$, then $\mathcal{A}^{DBDH}$ outputs 0 as its answer to the given instance of the DBDH problem. If $d' \neq d$, then $\mathcal{A}^{DBDH}$ outputs 1.

Let us denote as δ the probability that $\mathcal{A}_{\textsf{IND-CCA}}$ makes an unsigncryption query for a valid signcryption $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$ such that $\tilde{\mathsf{vk}} = \tilde{\mathsf{vk}}^\star$. In other words, δ is the probability that $\mathcal{A}^{DBDH}$ aborts before $\mathcal{A}_{\textsf{IND-CCA}}$ outputs its guess bit d'. Using a similar argument as in the unforgeability proof, it is easy to see that, in this case, one can construct a forger $\mathcal{F}_{\tilde{\Theta}}$ against the one-time signature scheme $\tilde{\Theta}$: the input of $\mathcal{F}_{\tilde{\Theta}}$ is $\tilde{\mathsf{vk}}^\star$, the only access to the signing oracle is used to compute the challenge signcryption, and any valid unsigncryption query coming from $\mathcal{A}_{\textsf{IND-CCA}}$ which involves $\tilde{\mathsf{vk}}^\star$ leads to a valid strong forgery of the signature scheme $\tilde{\Theta}$. Therefore, we have $\delta \leq \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda)$.

Let us now compute the probabilities that the output of the constructed solver $\mathcal{A}^{DBDH}$ of the DBDH problem is 0 in the two possible cases. When $R = e(g,g)^{abc}$, then the challenge signcryption is consistent, and we have $\Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0] = \delta \cdot \frac{1}{2} \ + \ (1-\delta) \cdot (\mathsf{Adv}_{\mathcal{A}_{\textsf{IND-CCA}}}(\lambda) + \frac{1}{2})$.

When $R = T$ is a random element in $\mathbb{G}_T$, the challenge signcryption is independent of the bit d, so the probability that $d' = d$ is 1/2, and we have $\Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, T) = 0] = \delta \cdot \frac{1}{2} \ + \ (1-\delta) \cdot \frac{1}{2}$.

Now we have $\mathtt{Adv}_{\mathcal{A}^{DBDH}}(\lambda) =$

$$\left| \Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0] \ - \ \Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, T) = 0] \right| =$$

$$= (1-\delta)\mathsf{Adv}_{\mathcal{A}_{\textsf{IND-CCA}}}(\lambda) = \mathsf{Adv}_{\mathcal{A}_{\textsf{IND-CCA}}}(\lambda) - \delta \cdot \mathsf{Adv}_{\mathcal{A}_{\textsf{IND-CCA}}}(\lambda).$$

Putting all together, we have, as desired:

$$\mathsf{Adv}_{\mathcal{A}_{\textsf{IND-CCA}}}(\lambda) = \mathtt{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \delta \cdot \mathsf{Adv}_{\mathcal{A}_{\textsf{IND-CCA}}}(\lambda) \leq$$

$$\leq \mathtt{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \delta = \mathtt{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda).$$

□

7 Efficiency of the Schemes

The two schemes proposed in this paper are the first PKI-based threshold unsigncryption schemes which achieve a high enough level of security. In particular, generic constructions obtained by composing a fully secure signature scheme and a fully secure threshold decryption scheme do not achieve this level of security, as we have shown in Section 4.

Therefore, our first goal was to show that the maximum level of security for threshold unsigncryption schemes can indeed be achieved. This is what we have done with our two proposals. Regarding efficiency, there are not previous schemes with the same level of security to compare with, so it is not possible to say if the two new schemes are efficient or not. Anyway, we include Table 1 that summarizes the computational and communication costs of our schemes (without considering robustness). The costs of these two schemes should be considered as a benchmark for any future proposal of threshold unsigncryption scheme.

Scheme	cost of Signcryption	$\lvert C\rvert$	cost of Unsigncryption (*per* receiver)	Security model
Section 5	6 Exp	6λ	8 Exp	ROM
Section 6	12 Exp + 1 Pa	12λ	11 Exp + 6 Pa	Standard

Table 1. Efficiency of our two threshold unsigncryption schemes.

To measure the efficiency of our second scheme, in Section 6, we have taken as the signature scheme Θ the scheme in [5], and as the one-time signature scheme $\tilde{\Theta}$ the scheme in Appendix B of [17]. In the table, λ denotes the security parameter of the scheme; this means that an element in the group $\mathbb{G}$ can be represented by λ bits. In the case of our first scheme, we have considered that the length of the plaintexts is $\ell = \lambda$, for simplicity.

We denote the size in bits of a ciphertext C as $|C|$. For the computational costs, we just consider exponentiations (denoted as Exp) and bilinear pairing computations (denoted as Pa, only for the second scheme). The rest of operations (xor, modular addition and multiplication, hash computations) are not considered because they are very cheap; they do not affect the real efficiency of the schemes. Roughly speaking, we can say that the scheme in Section 6, whose security is proved in the standard model, is twice less efficient than the scheme in Section 5, whose security is proved in the random oracle model (ROM).

8 Splitting the Unsigncryption Protocol

If we go back to the description of the Threshold Unsigncryption protocol of the two new schemes, in Sections 5 and 6, we can easily distinguish two parts in those protocols. Steps 1-3 correspond to the (public) verification procedure; these authentication steps can be run by any (individual) party, not necessarily inside the set B of intended receivers. In other words, no secret information is needed as input to run these three steps; the only inputs are the ciphertext and the public key of the sender. Then, Steps 4-7 correspond to the (secret) decryption procedure, which in this case requires the participation of at least t members of the set B of intended receivers. The important point here is that the identity or public key pk_A of the sender is not used at all for the execution of Steps 4-7. In some sense, the public key pk_A of the sender could be removed from the process once the ciphertext has been accepted as valid, in Step 3. After that, the identity of the sender would be unknown during the rest of the unsigncryption process.

In this way, the unsigncryption part of our two new schemes could be split into two parts. The first one could be run by an entity $T \notin B$, who would discard invalid ciphertexts and remove (or store privately) the identities of the senders. In the second part, only valid (and anonymized) ciphertexts would reach the set B of receivers, who would jointly decrypt the ciphertext to recover the original plaintexts, maybe without knowing at any moment who are the senders of the messages.

As far as we know, these are the first signcryption schemes (with either individual or threshold unsigncryption) enjoying this property, which may be of interest in some applications requiring some level of anonymity or privacy, such as electronic auctions.

In an electronic auction system, participants send their confidential and authenticated bids for a product. At the end of the process, some (distributed) entity B detects the highest bid and identifies the author of that bid, who wins the right to buy the product for that price. Identities of the authors of the rest of bids should remain hidden. To increase the confidentiality of the process, entity B can consist of a set of n entities, working in a (t, n)-threshold fashion.

Let us assume that such an auction system is implemented by using a signcryption scheme where the unsigncryption protocol can be split into two phases, in such a way that the decryption part is anonymized. An external authority (or machine) T, different from B, can be in charge of the first part of the unsigncryption process: T receives the ciphertexts from the participants in the auction, verifies that the participants are in the list of admitted participants, and runs the verification part of the unsigncryption. Invalid ciphertexts are discarded, and valid ciphertexts are anonymized and forwarded to the auction decryption entity B. Entity T must privately store a table (pk_A, C), relating the public keys of the participants with their ciphertexts. Optionally, the anonymized ciphertexts that are forwarded to B (or a hashed version of them) can be published so that all the participants in the auction can verify that their bids have been taken into account.

The decryption process is then run by entity B, in an individual or threshold fashion, and the highest bid among the resulting (anonymous) bids is selected. The winning bid and its corresponding ciphertext C are announce by B, and then T can search in its table and recover the identity of the author of the winning bid. Assuming the honesty of entity T, the anonymity of the participants that do not win the auction is clearly preserved, even in front of the decryption authority B. Since the role of T can be easily implemented by a secure piece of hardware, trusting entity T is not a very strong assumption.

9 Fully Threshold Signcryption

In this work we have considered, for simplicity, the scenario where the entity B that runs the unsigncryption process consists of a set of n individuals and works with a (t, n)-threshold mode of operation, but the entity that runs the signcryption process (the sender A) is an individual entity.

However, it is quite easy to see that our definitions and results (both the negative and the positive ones) extend to the scenario where the sender entity A also consists of a set of $\tilde{n}$ individuals and works with a $(\tilde{t}, \tilde{n})$-threshold process. Such a scenario can also make perfect sense in some real applications, for example in critical electronic auctions where an important public contract is put out to tender. A signed confidential bid on behalf of a company should require the participation of a minimum number of individuals in the board of the company.

The first threshold unsigncryption scheme that we propose, in Section 5, can be extended to this fully threshold scenario by using well-known threshold techniques for the computation of zero-knowledge proofs in the Discrete Logarithm framework (see [23] for the particular case of threshold Schnorr signatures, for instance). Regarding our second threshold unsigncryption scheme, in Section 6, the idea would be to replace the individual signature schemes Θ and $\tilde{\Theta}$ with threshold signature schemes. Examples of threshold signature schemes which are secure in the standard model can be found, for example, in [24, 15].

We point out that the two resulting fully threshold signcryption schemes would still enjoy the property discussed in the previous section: the unsigncryption protocol can be split into two parts.

10 Conclusion

We have considered in this paper the strong security properties that one could (or should) require for a signcryption scheme with threshold unsigncryption: existential unforgeability under insider chosen message attacks and indistinguishability under insider chosen ciphertext attacks, in a multi-user setting. Most of the (few) threshold unsigncryption schemes proposed in the literature, either in the traditional PKI or in the identity-based scenario, do not achieve this level of security. This includes generic constructions obtained by composing a fully secure signature scheme and a fully secure threshold decryption scheme.

We have constructed in this paper two threshold unsigncryption schemes which achieve those strong security properties. We prove the security of the first one in the random oracle model, whereas we are able to prove the security of the second proposed scheme in the standard model. The two schemes enjoy a "splitting" property which can be very useful for applications requiring some level of privacy for the sender of the digital information. As future work, one could investigate if other (more efficient) threshold unsigncryption schemes can be designed, with full security in the standard model, maybe without using bilinear maps.

References

1. J.H. An, Y. Dodis and T. Rabin. On the security of joint signature and encryption. *Proceedings of Eurocrypt'02*, LNCS **2332**, Springer-Verlag, pp. 83–107 (2002).
2. M. Bellare and P. Rogaway. Minimizing the use of random oracles in authenticated encryption schemes. *Proceedings of Information and Communications Security'97*, LNCS **1334**, Springer-Verlag, pp. 1–16 (1997).
3. D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 223–238 (2004).
4. D. Boneh, X. Boyen and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. *Proceedings of CT-RSA'06*, LNCS **3860**, Springer-Verlag, pp. 226–243 (2006).
5. D. Boneh, E. Shen and B. Waters. Strongly unforgeable signatures based on Computational Diffie-Hellman. *Proceedings of PKC'06*, LNCS **3958**, Springer-Verlag, pp. 229–240 (2006).
6. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Adaptive security for threshold cryptosystems. *Proceedings of Crypto'99*, LNCS **1666**, Springer-Verlag, pp. 98–115 (1999).
7. R. Canetti, S. Halevi and J. Katz. Chosen-ciphertext security from identity-based encryption. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 207–222 (2004).
8. R. Canetti, H. Krawczyk and J.B. Nielsen. Relaxing chosen-ciphertext security. *Proceedings of Crypto'03*, LNCS **2729**, Springer-Verlag, pp. 565–582 (2003).
9. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure distributed key generation for Discrete-Log based cryptosystems. *Journal of Cryptology*, vol. **20** (1), pp. 51–83 (2007).
10. J. Herranz, A. Ruiz and G. Sáez. Fully secure threshold unsigncryption. *Proceedings of ProvSec'10*, LNCS **6402**, Springer-Verlag, pp. 261–278 (2010).
11. S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: introducing concurrency, removing erasures. *Proceedings of Eurocrypt'00*, LNCS **1807**, Springer-Verlag, pp. 221–242 (2000).
12. J.H. Koo, H.J. Kim, I.R Jeong, D.H. Lee and J.I Lim. Jointly unsigncryptable signcryption schemes. *Proceedings of WISA'01*, vol. 2, pp. 397–407 (2001).
13. F. Li, J. Gao and Y. Hu. ID-based threshold unsigncryption scheme from pairings. *Proceedings of CISC'05*, LNCS **3822**, Springer-Verlag, pp. 242–253 (2005).
14. F. Li, X. Xin and Y. Hu. ID-based signcryption scheme with (t, n) shared unsigncryption. *International Journal of Network Security*, ovl. **3** (2), pp. 155–159 (2006).
15. J. Li, T.H. Yuen and K. Kim. Practical threshold signatures without random oracles. *Proceedings of ProvSec'07*, LNCS **4784**, Springer-Verlag, pp. 198–207 (2007).
16. A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: from cryptosystems to signature schemes. *Proceedings of Asiacrypt'01*, LNCS **2248**, Springer-Verlag, pp. 331–350 (2001).
17. P. Mohassel. One-time signatures and chameleon hash functions. *Proceedings of SAC'10*, LNCS **6544**, Springer-Verlag, pp. 302–319 (2011).
18. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, vol. **13** (3), pp. 361–396 (2000).
19. A. Shamir. How to share a secret. *Communications of the ACM*, vol. **22**, pp. 612–613 (1979).
20. S. Sharmila Deva Selvi, S. Sree Vivek, S. Priti and C. Pandu Rangan. On the security of identity based threshold unsigncryption schemes. *Proceedings of APWCS'2010*, available at `http://eprint.iacr.org/2010/360` (2010).
21. V. Shoup. Lower bounds for discrete logarithms and related problems. *Proceedings of Eurocrypt'97*, LNCS **1233**, Springer-Verlag, pp. 256–266 (1997).
22. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, vol. **15** (2), pp. 75–96 (2002).
23. D.R. Stinson and R. Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. *Proceedings of ACISP'01*, LNCS **2119**, Springer-Verlag, pp. 417–434 (2001).
24. H. Wang, Y. Zhang and D. Feng. Short threshold signature schemes without random oracles. *Proceedings of Indocrypt'05*, LNCS **3797**, Springer-Verlag, pp. 297–310 (2005).
25. B. Yang, Y. Yu, F. Li and Y. Sun. Provably secure identity-based threshold unsigncryption scheme. *Proceedings of ATC'07*, LNCS **4610**, Springer-Verlag, pp. 114–122 (2007).

26. Z. Zhang, C. Mian and Q. Jin. Signcryption scheme with threshold shared unsigncryption preventing malicious receivers. *Proceedings of TENCON'02*, IEEE Computer Society, vol. 2, pp. 196–199 (2002).
27. Y. Zheng. Digital signcryption or How to achieve cost(signature & encryption) << cost(signature) + cost(encryption). *Proceedings of Crypto'97*, LNCS **1294**, Springer-Verlag, pp. 165–179 (1997).