SIGNCRYPTION WITH NON-INTERACTIVE NON-REPUDIATION

JOHN MALONE-LEE

ABSTRACT. Signcryption [35] is a public key primitive that achieves the functionality of both an encryption scheme and a signature scheme simultaneously. It does this more efficiently than a composition of public key encryption and public key signature.

We present a model of security for signcryption schemes that offer *non-interactive non-repudiation*. This is non-repudiation in which the judge settling a repudiation dispute does not have to get involved in an interactive zero-knowledge proof. Our model applies to many existing schemes in the literature [5, 23, 30].

We explain why the scheme proposed in [5] is insecure under any definition of privacy based on the idea of indistinguishable encryptions [21]. We describe a modified scheme to overcome the problem. Proofs of security are given for the scheme in the random oracle model [11].

1. INTRODUCTION

Signcryption is a novel public key primitive first proposed by Zheng in 1997 [35]. A signcryption scheme combines the functionality of a digital signature scheme with that of an encryption scheme. It therefore offers the three services: privacy, authenticity and non-repudiation. Since these services are frequently required simultaneously, Zheng proposed signcryption as a means to offer them in a more efficient manner that a straightforward composition of digital signature scheme and encryption scheme. The title of [35] stated that it should be possible to achieve

$$\text{(1)} \quad \mathrm{Cost}(\mathrm{Signature}\ \&\ \mathrm{Encryption}) << \mathrm{Cost}(\mathrm{Signature}) + \mathrm{Cost}(\mathrm{Encryption}),$$

and an ingenious scheme was proposed to meet such a goal.

Since the primitive was proposed, many signcryption schemes have been designed [5, 23, 25, 30]. These schemes have all been constructed without a precisely specified model of security and a corresponding proof. Recently however several formal models of security for signcryption schemes have emerged [2, 3, 4]. In [4] a variant of the original signcryption scheme from [35] is proved secure in the random oracle model [11]. The model in [3] differs from the original idea in two respects. First of all it does not make non-repudiation a requirement of signcryption. Secondly it does not require (1) to hold. The goal is simply to analyse primitives that achieve the combined functionality of signature and encryption. This includes signcryption schemes such as [35] but it could also include compositions of signature schemes and encryption schemes.

The original scheme proposed in [35], and the variant of it in [4], concentrated on achieving privacy and authenticity in a very efficient manner. These schemes are remarkable in that signcryption requires only one group exponentiation. In [4] a proof of security is given for the scheme in a model where these are the objectives. The only disadvantage of the scheme is the way it achieves non-repudiation. For an untrusted trusted judge to settle a repudiation dispute the zero-knowledge proofs of [10, 14] are required.

In [5] a modified version of Zheng's original signcryption scheme is proposed that offers non-repudiation in a more straightforward manner. We will explain why this scheme is insecure under any definition of privacy based on indistinguishability of encryptions (or indistinguishability of signcryptions in this case). We will also propose a new scheme that overcomes this weakness.

Non-repudiation for signcryption is not a straightforward consequence of unforgeability as it is for digital signature schemes. The reason for this is that a signcrypted message is "encrypted" as well as "signed". Therefore, by default, only the intended receiver of a signcryption may verify its authenticity. If a third party is to settle a repudiation dispute over a signcryption it must have access to some information other than the signcryption itself. It is observed in [30] that one of the non-repudiation procedures suggested for Zheng's original signcryption has a weakness: information given to a third party to settle a dispute compromises the privacy offered by the scheme.

The aim of this work is twofold. Firstly we propose a model of security for signcryption schemes that offer non-interactive non-repudiation with an untrusted judge. In this context non-interactive refers to the fact that the judge is a passive verifier. It does not to have to get involved in an interactive zero-knowledge proof to settle a repudiation dispute as is the case for the scheme in [35]. The second aim is to consider signcryption in a multi-user setting similar to that of [7] for public key encryption and [20] for public key signature schemes. Our model differs from that of [3] in that we allow the adversary to choose which users it attacks. We give proofs of security for a scheme similar to that of [5]. Our proofs show how the security of our scheme is preserved as the numbers of users of the scheme increases.

The remainder of this paper is organised as follows. In Section 2 we define precisely the primitive that we are interested in. Once we have done this we give details of a scheme satisfying our definition. We present security definitions in Section 3. Our security results are given in Section 4. We finish with some concluding remarks.

2. SIGNCRYPTION WITH NON-INTERACTIVE NON-REPUDIATION

This paper is concerned with signcryption schemes that have a non-repudiation procedure of a particularly simple form. We will call the primitive signcryption with non-interactive non-repudiation (SCNINR). It is described formally in Definition 1 below.

Before presenting the definition, let us give some notation that will be used throughout. If S is a set then $x \stackrel{r}{\leftarrow} S$ denotes the algorithm that selects an element uniformly at random from S and assigns the outcome to x. Similarly, if A is a probabilistic algorithm $x \stackrel{r}{\leftarrow} A(x_1, \ldots, x_n)$ denotes assigning to x the output of A on inputs $x_1, \ldots, x_n$ and some input drawn uniformly at random from an appropriate set. We denote assignment of y to x by $x \leftarrow y$.

Definition 1 (Signcryption with non-interactive non-repudiation)**.** *A signcryption scheme with non-interactive non-repudiation consists of six algorithms* $(\mathcal{SP}, \mathcal{K}, \mathcal{S}, \mathcal{U}, \mathcal{N}, \mathcal{PV})$.

- *The system parameters generation algorithm* $\mathcal{SP}$ *is randomised. It takes as input a security parameter* 1^k *and returns some global information* I*. We write* $I \stackrel{r}{\leftarrow} \mathcal{SP}(1^k)$.
- *The user key generation algorithm* $\mathcal{K}$ *is randomised. It takes as input global information* I *and returns a matching secret and public key pair* (x, Y)*. We write* $(x, Y) \stackrel{r}{\leftarrow} \mathcal{K}(I)$.
- *The signcryption algorithm* $\mathcal{S}$ *is randomised. It takes as input a sender's secret key* x_i^a*, a sender's public key* Y_i^a*, a receiver's public key* Y_j^b *and a plaintext* m*. It returns a ciphertext* σ*. We write* $\sigma \stackrel{r}{\leftarrow} \mathcal{S}_{\langle x_i^a, Y_i^a, Y_j^b \rangle}(m)$.

- *The unsigncryption algorithm $\mathcal{U}$ is deterministic. It takes as input a sender's public key Y_i^a, a receiver's secret key x_j^b, a receiver's public key Y_j^b and a string σ. It returns either a message m, or the distinguished symbol $\perp$. We write $x \leftarrow \mathcal{U}_{\langle Y_i^a, x_j^b, Y_j^b \rangle}(\sigma)$, where x is either a message m or $\perp$. The symbol $\perp$ indicates that σ is not a valid signcryption.*
- *The non-repudiation algorithm $\mathcal{N}$ takes as input a sender's public key Y_i^a, a receiver's secret key x_j^b, a receiver's public key Y_j^b and a string σ. If σ is a valid signcryption it returns keys Y_i^a, Y_j^b, a message m and a string ι. The string ι is information that makes public verification that the message m was sent by the owner of Y_i^a to the owner of Y_j^b possible. That is to say, from a signcryption σ, $\mathcal{N}$ extracts a digital signature under Y_i^a on m. The public verification algorithm $\mathcal{PV}$ below is then used to verify the signature. If σ is an invalid signcryption $\mathcal{N}$ returns $\perp$.*
- *The public verification algorithm $\mathcal{PV}$ takes as input a sender's public key Y_i^a, a receiver's public key Y_j^b a message m and purported signature ι (as output by $\mathcal{N}$ above). It returns either $\top$ or $\perp$ depending on whether or not ι is a valid signature. Note that this provides non-repudiation of origin but not of receipt.*

We require that, for all keys returned by $\mathcal{K}(I)$, and all messages $m \in \{0,1\}^*$ we have $\mathcal{U}_{\langle Y_i^a, x_j^b, Y_j^b \rangle}(\mathcal{S}_{\langle x_i^a, Y_i^a, Y_j^b \rangle}(m)) = m$. We assume that the security parameter is available to all algorithms after being explicitly provided to $\mathcal{SP}$ and that these all run in polynomial time (in the security parameter).

Several schemes fitting Definition 1 already exist in the literature. One example is the original scheme of [35] when used with a particular non-repudiation procedure. This procedure involves surrendering the ephemeral key used for encryption together with a zero-knowledge argument that the key has the correct form. Using a protocol of [16] a non-interactive argument can be given to prove that the ephemeral key has the correct form. This, together with the ephemeral key itself, would then correspond to ι in Definition 1. The problem with this approach is that it compromises the privacy of messages sent by the sender in question to the receiver in question. This is discussed in [30] where a modified scheme is suggested to overcome this problem, this scheme also fits Definition 1. Other existing schemes fitting Definition 1 are given in [5, 23].

Note that we are not claiming that all signcryption schemes have the form of Definition 1: our aim is to provide a framework in which to formally analyse those that do.

We will give a concrete example of such a scheme constructed in a hybrid manner using a symmetric encryption scheme. The scheme is a modification of that in [5]. As we will discuss below, the actual scheme of [5] does not have indistinguishable signcryptions, even under a passive attack, our modification overcomes this problem.

Our scheme uses a group for which there exists a non-degenerate, bilinear, computable map to a second group. This choice is not essential for the functionality of the scheme; however, using such a group simplifies our proofs and allows a tighter reduction in the case of privacy. It would be possible to give similar proofs for more general groups under the Gap Diffie-Hellman assumption as in [4]. There is a quadratic degradation in the time complexity for the proof of privacy using this method however.

Definition 2 (Non-degenerate, bilinear, computable map)**.** *Let G and G' be cyclic groups of prime order q, where G is additive and G' is multiplicative. Let $e : G \times G \to G'$ be a map with the properties below.*

(1) *Non-degenerate: There exists $X, Y \in G$ such that $e(X, Y) \neq 1$.*

(2) *Bilinear:*

$$e(X_1 + X_2, Y) = e(X_1, Y) \cdot e(X_2, Y) \text{ and } e(X, Y_1 + Y_2) = e(X, Y_1) \cdot e(X, Y_2)$$

(3) *Computable: There is an efficient algorithm for evaluating e.*

Groups with such a map may be derived from subgroups of elliptic curve groups $E(\mathbb{F}_q)$, whose order r divides $q^k - 1$ but does not divide $q^i - 1$ for $0 < i < k$, and r is large enough to resist the MOV attack [27], but small enough to make computing the Tate or Weil pairing feasible. A modified version of the Tate or Weil pairing may be used with some of these groups to give the required map. This technique was first described in [34] and may be used for supersingular elliptic curves. Recently these groups have been used constructively to build identity-based cryptosystems [13, 15, 24, 26, 29, 33]. As a result there has been much interest in efficient implementation of the Tate pairing [6, 19].

Finally, before describing the scheme, we define what we mean by a symmetric encryption scheme.

Definition 3 (Symmetric encryption scheme)**.** *A symmetric encryption scheme $\mathcal{SE}$ consists of three algorithms $\mathcal{SE} = (\mathcal{K}_{se}, \mathcal{E}, \mathcal{D})$.*

- *The key generation algorithm $\mathcal{K}_{se}$ is a randomised algorithm that takes as input a security parameter 1^k and returns a key κ. We write $\kappa \stackrel{r}{\leftarrow} \mathcal{K}_{se}(1^k)$ and we denote the set of all strings that have non-zero probability of being output by $\mathcal{K}_{se}(1^k)$ as $Ke(\mathcal{SE})$.*
- *The encryption algorithm $\mathcal{E}$ may be a randomised or deterministic algorithm. It takes as input a key $\kappa \in Ke(\mathcal{SE})$ and a plaintext $m \in \{0,1\}^*$. It returns a ciphertext $c \in \{0,1\}^*$. We write $c \stackrel{r}{\leftarrow} \mathcal{E}_\kappa(m)$.*
- *The decryption algorithm $\mathcal{D}$ is a deterministic algorithm that takes as input a key $\kappa \in Ke(\mathcal{SE})$ and a ciphertext $c \in \{0,1\}^*$. It returns a plaintext $m \in \{0,1\}^*$. We write $m \leftarrow \mathcal{D}_\kappa(c)$.*

It is required that, for any $\kappa \in Ke(\mathcal{SE})$, and any $m \in \{0,1\}^*$, we have $\mathcal{D}_\kappa(\mathcal{E}_\kappa(m)) = m$.

We are now ready to give a concrete example of a signcryption scheme with non-interactive non-repudiation. We call our scheme SC and describe it in Definition 4 below.

Definition 4 (SC)**.** *Let $\mathcal{SE} = (\mathcal{K}_{se}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with $Ke(\mathcal{SE}) = \{0,1\}^{l_e}$. Let $H_1 : \{0,1\}^* \to \{0,1\}^{l_e} \times \{0,1\}^{l_h}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ be hash functions. Based on these primitives the signcryption scheme $SC = (\mathcal{SP}, \mathcal{K}, \mathcal{S}, \mathcal{U}, \mathcal{N}, \mathcal{PV})$ is constructed as follows:*

Algorithm $\mathcal{SP}(1^k)$
Select a finite Abelian groups G and G' of prime order q where
$|q| = k$ ($|q|$ denotes the bit length of q) such that there is
a non-degenerate, bilinear, computable map $e : G \times G \to G'$
Select a generator P of G and let $\mathcal{O}$ denote the identity element
Let I be a description of $e, G, G', P, q, \mathcal{SE}, H_1$ and H_2
Return I

Algorithm $\mathcal{K}(I)$
$x \xleftarrow{r} \mathbb{Z}_q^*$
$Y \leftarrow xP$
Return (x, Y)

Algorithm $\mathcal{S}_{\langle x_i^a, Y_i^a, Y_j^b \rangle}(m)$
$z \xleftarrow{r} \mathbb{Z}_q^*$
$U \leftarrow zY_j^b$
$V \leftarrow zP$
$\kappa_1 || \kappa_2 \leftarrow H_1(U)$
$c \leftarrow \mathcal{E}_{\kappa_1}(m)$
$r \leftarrow H_2(V||m||\kappa_2||Y_i^a||Y_j^b)$
$s \leftarrow z - x_i^a \cdot r \mod q$
$\sigma \leftarrow (c, r, s)$
Return σ

Algorithm $\mathcal{U}_{\langle Y_i^a, x_j^b, Y_j^b \rangle}(\sigma)$
Parse σ *as* (c, r, s)
Verify $r \in \mathbb{Z}_q^* \wedge s \in \mathbb{Z}_q^*$,
if not return $\perp$
$V \leftarrow sP + rY_i^a$
If $V = \mathcal{O}$, *return* $\perp$
$U \leftarrow x_j^b V$
$\kappa_1 || \kappa_2 \leftarrow H_1(U)$
$m \leftarrow \mathcal{D}_{\kappa_1}(c)$
If $r \neq H_2(V||m||\kappa_2||Y_i^a||Y_j^b)$,
return $\perp$
Return m

Algorithm $\mathcal{N}_{\langle Y_i^a, x_j^b, Y_j^b \rangle}(\sigma)$
Parse σ *as* (c, r, s)
Verify $r \in \mathbb{Z}_q^* \wedge s \in \mathbb{Z}_q^*$,
if not return $\perp$
$V \leftarrow sP + rY_i^a$
If $V = \mathcal{O}$, *return* $\perp$
$U \leftarrow x_j^b V$
$\kappa_1 || \kappa_2 \leftarrow H_1(U)$
$m \leftarrow \mathcal{D}_{\kappa_1}(c)$
If $r \neq H_2(V||m||\kappa_2||Y_i^a||Y_j^b)$,
return $\perp$
$\iota \leftarrow (\kappa_2, r, s)$
Return Y_i^a, Y_j^b, m, ι

Algorithm $\mathcal{PV}_{\langle Y_i^a, Y_j^b \rangle}(m, \iota)$
Parse ι *as* (κ_2, r, s)
Verify $\kappa_2 \in \{0,1\}^{l_h} \wedge r \in \mathbb{Z}_q^* \wedge s \in \mathbb{Z}_q^*$,
if not return $\perp$
$V \leftarrow sP + rY_i^a$
If $V = \mathcal{O}$, *return* $\perp$
If $r \neq H(V||m||\kappa_2||Y_i^a||Y_j^b)$, *return* $\perp$
Return $\top$

One of the most significant difference between our scheme and that of [5] is that our scheme includes κ_2 as input to the hash function H_2. This is crucial for the security of the scheme. Consider what would happen if we had a similar scheme, the only difference being that κ_2 was no longer included in the H_2 input. Suppose that an adversary is given (c, r, s): a signcryption of either m_0 or m_1 created by the owner of Y_i^a for the owner of Y_j^b. The adversary can compute $V = sP + rY_i^a$ and check whether or not $r = H_2(V||m_0||Y_i^a||Y_j^b)$ i.e. it can determine by simple elimination which message has been signcrypted. As we discuss in Section 3.1 below, this violates the generally accepted definition of privacy for any scheme offering encryption. It is easily verified that the scheme of [5] also suffers from this major weakness. We use κ_2 to prevent this trivial attack. Setting $l_h \approx 60$ should suffice.

Note that, once $Y_i^a, Y_j^b, m, (\kappa_2, r, s)$ is released by $\mathcal{N}$ algorithm in Definition 4, the algorithm $\mathcal{PV}$ is just the verification process for the signature scheme of Schnorr [31, 32].

3. Security notions for SCNINR

3.1. **Privacy of SCNINR in a Multi-User Setting.** Like [7] for public key encryption, and [20] for signature schemes, we consider the privacy of SCNINR schemes in a network of n users. The goal of an adversary here is to determine which of two chosen messages has been signcrypted. This idea originates in [21] where the analogous notion was dubbed *semantic security*. Since then it has become the underpinning idea in the definitions of security for all types of encryption scheme [1, 3, 4, 8, 9, 12, 17, 18]. We describe the particular attack scenario that we are interested in below.

To begin with global parameters are generated. Using these global parameters two key pairs are generated for each of the n users. We say that a user has a *sender* key pair and a *receiver* key pair. For user i these are denoted (x_i^a, Y_i^a) and (x_i^b, Y_i^b) respectively. Which key is used depends on the role of a user in a particular communication: sender or receiver. Referring to Definition 4, for user i to send a message to user j one would signcrypt/unsigncrypt as follows: $\mathcal{S}_{\langle x_i^a, Y_i^a, Y_j^b \rangle}(m)/\mathcal{U}_{\langle Y_i^a, x_j^b, Y_j^b \rangle}(\sigma)$.

An adversary will operate in two stages. In the first, or find, stage it is given the global parameters and the n pairs of public keys of the users. Similar to an adaptive chosen ciphertext attack on a public key encryption scheme [1, 9, 12, 17, 18], the adversary is provided with an unsigncryption oracle that it may call for any pair of users. The adversary may also query a signcryption oracle that will signcrypt any message for any pair of users. We must provide an adversary with such an oracle since, unlike the case of a public key encryption scheme, an adversary is not able to signcrypt messages itself. This means that we are proving security in a model where the adversary is assumed to be outside the network. Such a model was dubbed *outsider security* in [3]. Unfortunately our scheme does not offer privacy against insiders: referring to Definition 4, a user who knows r, s and x_i^a can easily recover the ephemeral value z used to generate the encryption key.

The final oracle given to the adversary is the one used to generate the publicly verifiable information necessary for non-repudiation. We say that this is the *non-repudiation oracle*. The adversary may call this oracle with the appropriate public keys of any pair of users and any string. We provide this oracle for two reasons. First of all, if the adversary is able to gain access to the signcryption oracle and unsigncryption oracle, there is no reason that it should not be able to access this oracle. Secondly, as observed in [30], it is possible that information from such an oracle may compromise the privacy of the scheme.

At the end of the find stage the adversary outputs a pair of users for which it wants to be challenged: a sender and a receiver. One of the messages is chosen at random and signcrypted under the appropriate keys for the chosen sender and receiver. The resulting signcryption is called the target ciphertext.

In the second, or guess, stage the adversary is given the target ciphertext. Using identical oracles to those in the find stage the adversary must determine which of the two messages was signcrypted. There is the obvious restriction that the adversary may not query the unsigncryption oracle or the non-repudiation oracle with its chosen keys and the target ciphertext.

We describe this attack formally in Definition 5 below.

Definition 5 (Privacy of SCNINR)**.** *Let $\mathcal{SC} = (\mathcal{SP}, \mathcal{K}, \mathcal{S}, \mathcal{U}, \mathcal{N}, \mathcal{PV})$ be a SCNINR scheme. Let A_{cca} be an adversary that outputs a bit d. The adversary A_{cca} operates in two stages, the find stage A_f, and the guess stage A_g. In both stages of the attack the adversary has access to a signcryption oracle $\mathcal{S}_{\mathcal{O}}(\cdot,\cdot,\cdot)$. This oracle takes public keys Y_i^a, Y_j^b and a message m as input. It returns $\sigma \stackrel{r}{\leftarrow} \mathcal{S}_{\langle x_i^a, Y_i^a, Y_j^b \rangle}(m)$. The adversary has access to an unsigncryption oracle $\mathcal{U}_{\mathcal{O}}(\cdot,\cdot,\cdot)$. This oracle takes public keys Y_i^a, Y_j^b and a signcryption σ as input. It returns $\mathcal{U}_{\langle Y_i^a, x_j^b, Y_j^b \rangle}(\sigma)$. The adversary also has access to a non-repudiation oracle $\mathcal{N}_{\mathcal{O}}(\cdot,\cdot,\cdot)$. This oracle takes as input public keys Y_i^a, Y_j^b and a signcryption σ. It returns $\mathcal{N}_{\langle Y_i^a, x_j^b, Y_j^b \rangle}(\sigma)$. We denote these oracles $\mathcal{S}_{\mathcal{O}}, \mathcal{U}_{\mathcal{O}}$ and $\mathcal{N}_{\mathcal{O}}$ respectively below. Let 1^k be the security parameter. Consider the following experiment.*

Experiment $\mathbf{Exp}^{ind-cca}_{\mathcal{SC},A_{cca}}(n, 1^k)$
$I \xleftarrow{r} \mathcal{SP}(1^k)$
For $i = 1, \ldots, n$: $(x_i^a, Y_i^a) \xleftarrow{r} \mathcal{K}(I)$; $(x_i^b, Y_i^b) \xleftarrow{r} \mathcal{K}(I)$
$(m_0, m_1, A, B, state) \leftarrow A_f^{\mathcal{S}_\mathcal{O}, \mathcal{U}_\mathcal{O}, \mathcal{N}_\mathcal{O}}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
$d \xleftarrow{r} \{0, 1\}$
$\sigma^* \xleftarrow{r} \mathcal{S}_{\langle x^s_A, Y^a_A, Y^b_B\rangle}(m_d)$
$d' \leftarrow A_g^{\mathcal{S}_\mathcal{O}, \mathcal{U}_\mathcal{O}, \mathcal{N}_\mathcal{O}}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b, \sigma^*, m_0, m_1, state)$
If $d' = d$ *return 1, else return 0*

It is mandated that the m_0 *and* m_1 *output by* A_f *are of equal length and that* A_g *never queries the oracle* $\mathcal{U}_\mathcal{O}$*, or the oracle* $\mathcal{N}_\mathcal{O}$*, with* Y_A^a, Y_B^b *and* σ^**.*

We define the advantage of the adversary as

$$\mathbf{Adv}^{ind-cca}_{\mathcal{SC},A_{cca}}(n, 1^k) = 2 \cdot \Pr[\mathbf{Exp}^{ind-cca}_{\mathcal{SC},A_{cca}}(n, 1^k) = 1] - 1.$$

For any integers $t, q_s, q_u, q_n \geq 0$*, we define the advantage function of the scheme*

$$\mathbf{Adv}^{ind-cca}_{\mathcal{SC}}(n, 1^k, t, q_s, q_u, q_n) = \max_{A_{cca}}\{\mathbf{Adv}^{ind-cca}_{\mathcal{SC},A_{cca}}(n, 1^k)\},$$

where the maximum is over all adversaries with time complexity t*, each making at most* q_s *queries to* $\mathcal{S}_\mathcal{O}(\cdot,\cdot,\cdot)$*, making at most* q_u *queries to* $\mathcal{U}_\mathcal{O}(\cdot,\cdot,\cdot)$*, and making at most* q_n *queries to* $\mathcal{N}_\mathcal{O}(\cdot,\cdot,\cdot)$*.*

The scheme $\mathcal{SC}$ *is said to be IND-CCA secure in the* n *user setting if the advantage function* $\mathbf{Adv}^{ind-cca}_{\mathcal{SC}}$ $(n, 1^k, t, q_s, q_u, q_n)$ *is a negligible function*[1] *of the security parameter. Note that this definition of IND-CCA security should not be confused with IND-CCA security for public key encryption schemes.*

3.2. **Unforgeability of SCNINR.** We take as our starting point the definition of security for public key signature scheme in the multi-user setting [20]. Several modifications to the definition are necessary before it can be used for signcryption schemes. We discuss these modifications before giving the definition formally.

Consider unforgeability in a network of n users. To begin with global parameters are generated. Using these global parameters a sender key pair and a receiver key pair are generated for each of the n user as described in Section 3.1.

The adversary is given access to three oracles. First of all, analogous to an adaptive chosen message attack on a signature scheme [20, 22], it has access to a signcryption oracle. It may call this oracle with a sending public key, an receiving public key and any message. When using a signature scheme, public verification of signatures is possible. This is not the case with a signcryption scheme and so we provide the adversary with an unsigncryption oracle that it may call with a sending public key, and receiving public key and any string. Like the definition of privacy, we also give the adversary a non-repudiation oracle that it may call with a sending public key, a receiving public key and any string. A proof in this model ensures that the information given away for public verification does not compromise the unforgeability of the scheme.

When using a signature scheme, the only private key used in signature generation belongs to the sender. An adversary can therefore be anyone, since there is no difference in the ability to forge signatures between a receiver of signed messages and an eavesdropping third party. For a SCNINR scheme however, signature generation uses the receiver's public key as well as the sender's keys. In this instance there may be a difference in the ability to

[1] A function $\epsilon(k)$ is *negligible* if for every c there exists a k_c such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_c$.

forge signcryptions between the receiver and a third party, since only the receiver knows the secret key corresponding to its public key. With this in mind we allow the adversary to come up with its own public key i.e. it becomes the $n+1$-th user in the network. The goal of the adversary is to come up with a valid forgery from any of the original n users to any user. That is to say that the adversary wins if it forges a signcryption from any of the original n user to itself or to any other user.

Note that, unlike the case of privacy dealt with in Section 3.1, we are considering unforgeability by a user inside the network. Such a scenario is treated in [3] where it is dubbed *insider security*. This is crucial if our scheme is to offer non-repudiation: it must be clear to a third party whether the sender or receiver produced a signcryption.

We give the whole definition formally below.

Definition 6 (Unforgeability of SCNINR)**.** *Suppose that $\mathcal{SC} = (\mathcal{SP}, \mathcal{K}, \mathcal{S}, \mathcal{U}, \mathcal{N}, \mathcal{PV})$ is a SCNINR scheme used in an n-user setting. Let A be an adversary whose goal is to produce a forgery. This forgery may be from one of the n users to another of the n users. Alternatively, the adversary may come up with its own public key Y_B and produce a forgery from one of the n users to itself.*

The adversary has access to a signcryption oracle $\mathcal{S}_{\mathcal{O}}(\cdot,\cdot,\cdot)$. This oracle takes as input public keys Y_i^a, Y_j^b and a message m. It returns $\sigma \xleftarrow{r} \mathcal{S}_{\langle x_i^a, Y_i^a, Y_j^b\rangle}(m)$. The adversary also has access to an unsigncryption oracle $\mathcal{U}_{\mathcal{O}}(\cdot,\cdot,\cdot)$. This oracle takes as input public keys Y_i^a, Y_j^b and a string σ. It returns $\mathcal{U}_{\langle Y_i^a, x_j^b, Y_j^b\rangle}(\sigma)$. The final oracle given to the adversary is a non-repudiation oracle $\mathcal{N}_{\mathcal{O}}(\cdot,\cdot,\cdot)$. This oracle takes as input public keys Y_i^a, Y_j^b and a string σ. It returns $\mathcal{N}_{\langle Y_i^a, x_j^b, Y_j^b\rangle}(\sigma)$. The attack is described in the experiment below where these oracles are denoted $\mathcal{S}_{\mathcal{O}}$, $\mathcal{U}_{\mathcal{O}}$ and $\mathcal{N}_{\mathcal{O}}$ respectively.

Experiment $\mathbf{Exp}^{cma}_{\mathcal{SC},A}(n, 1^k)$
$I \xleftarrow{r} \mathcal{SP}(1^k)$
For $i = 1, \ldots, n:\ (x_i^a, Y_i^a) \xleftarrow{r} \mathcal{K}(I)$; $(x_i^b, Y_i^b) \xleftarrow{r} \mathcal{K}(I)$
If $A^{\mathcal{S}_{\mathcal{O}}, \mathcal{U}_{\mathcal{O}}, \mathcal{N}_{\mathcal{O}}}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$ *outputs* Y_i^a, Y_B, m^*, ι^* *such that:*
1. $\top \leftarrow \mathcal{PV}_{\langle Y_i^a, Y_B\rangle}(m^*, \iota^*)$ *and,*
2. *the query* $\mathcal{S}_{\mathcal{O}}(Y_i^a, Y_B, m^*)$ *was never made,*
   (note that we may have $Y_B \notin \{Y_1^b, \ldots, Y_n^b\}$*)*

return 1, else return 0.

We define the advantage of the adversary as

$$\mathbf{Adv}^{cma}_{\mathcal{SC},A}(n, 1^k) = \Pr[\mathbf{Exp}^{cma}_{\mathcal{SC},A}(n, 1^k) = 1].$$

For any integers $t, q_s, q_u, q_n \geq 0$, we define the advantage function of the scheme

$$\mathbf{Adv}^{cma}_{\mathcal{SC}}(n, 1^k, t, q_s, q_u, q_n) = \max_A\{\mathbf{Adv}^{cma}_{\mathcal{SC},A}(n, 1^k)\},$$

where the maximum is over all adversaries with time complexity t, each making at most q_s queries to $\mathcal{S}_{\mathcal{O}}(\cdot,\cdot,\cdot)$, at most q_u queries to $\mathcal{U}_{\mathcal{O}}(\cdot,\cdot,\cdot)$ and at most q_n queries to $\mathcal{N}_{\mathcal{O}}(\cdot,\cdot,\cdot)$.

The scheme $\mathcal{SC}$ said to be existentially unforgeable against adaptive chosen message attack if its advantage function is a negligible function of the security parameter.

4. Security of SC in the multi-user setting

4.1. **Privacy of the scheme SC in the multi-user setting.** In this section we will prove SC to be secure under Definition 5. The security proof will be relative to the Computational Diffie-Hellman (CDH) assumption. We give a precise statement of this assumption below.

Definition 7 (The Computational Diffie-Hellman Assumption)**.** *Let* 1^k *be a security parameter and let* A *be an algorithm. Consider the experiment below.*

Experiment $\mathbf{Exp}^{cdh}_{\mathcal{SP},A}(1^k)$
Run $\mathcal{SP}(1^k)$ *from Definition 4 of* SC *to obtain* I
$x, y \xleftarrow{r} \mathbb{Z}_q^*$
$X \leftarrow xP, Y \leftarrow yP$
$Z \leftarrow A(1^k, I, X, Y)$
If $Z = xyP$ *return 1, else return 0*

We define the advantage of A *in solving CDH as*

$$\mathbf{Adv}^{cdh}_{\mathcal{SP},A}(1^k) = \Pr[\mathbf{Exp}^{cdh}_{\mathcal{SP},A}(1^k) = 1].$$

We define the advantage function as

$$\mathbf{Adv}^{cdh}_{\mathcal{SP}}(1^k, t) = \max_A\{\mathbf{Adv}^{cdh}_{\mathcal{SP},A}(1^k)\},$$

where the maximum is over all algorithms with time complexity t*. The CDH assumption is that* $\mathbf{Adv}^{cdh}_{\mathcal{SP}}(1^k, t)$ *is a negligible function of* 1^k*.*

We also require an assumption about the symmetric encryption scheme $\mathcal{SE}$ used in the construction of SC. The assumption is indistinguishability of encryptions under chosen plaintext attack (IND-CPA), as used in [1, 4]. The definition is given formally below.

Definition 8 (IND-CPA for Symmetric Encryption)**.** *Let* $\mathcal{SE} = (\mathcal{K}_{se}, \mathcal{E}, \mathcal{D})$ *be a symmetric encryption scheme as in Definition 3. Let* $A_{se} = (A_{se_1}, A_{se_2})$ *be an adversary that runs in two stages:* A_{se_1} *the find stage and* A_{se_2} *the guess stage. Consider the experiment below.*

Experiment $\mathbf{Exp}^{ind-cpa}_{\mathcal{SE},A_{se}}(1^k)$
$\kappa \xleftarrow{r} \mathcal{K}_{se}(1^k)$
$(m_0, m_1, state) \leftarrow A_{se_1}^{\mathcal{E}_\kappa(\cdot)}(1^k)$
$d \xleftarrow{r} \{0,1\}$
$c^* \xleftarrow{r} \mathcal{E}_\kappa(m_d)$
$d' \leftarrow A_{se_2}^{\mathcal{E}_\kappa(\cdot)}(1^k, c^*, m_0, m_1, state)$
If $d' = d$ *return 1, else return 0*

In the above experiment $\mathcal{E}_\kappa(\cdot)$ *denotes the oracle that, when queried with* $m \in \{0,1\}^*$*, returns* $c \xleftarrow{r} \mathcal{E}_\kappa(m)$*. It is mandated that* $|m_0| = |m_1|$*.*

We define the advantage of A_{se} *as*

$$\mathbf{Adv}^{ind-cpa}_{\mathcal{SE},A_{se}}(1^k) = 2 \cdot \Pr[\mathbf{Exp}^{ind-cpa}_{\mathcal{SE},A_{se}}(1^k) = 1] - 1.$$

For any 1^k*,* t *and* q_e *we define the advantage function of* $\mathcal{SE}$ *as*

$$\mathbf{Adv}^{ind-cpa}_{\mathcal{SE}}(1^k, t, q_e) = \max_{A_{se}}\{\mathbf{Adv}^{ind-cpa}_{\mathcal{SE},A_{se}}(1^k)\},$$

where the maximum is over all adversaries with time complexity t*, each making at most* q_e *queries to* $\mathcal{E}_\kappa(\cdot)$*. We consider* $\mathcal{SE}$ *to be IND-CPA secure if its advantage function is a negligible function of the security parameter.*

Using Definition 7 and Definition 8 we are now ready to state and prove our privacy result for SC.

FIGURE 1

$$
\begin{array}{l}
\text{Algorithm } B_{cdh}(1^k, I, X, Y) \\
\quad \text{For } i = 1, \ldots, n: \\
\quad\quad \dagger\ r_i^*, s_i^* \leftarrow \mathbb{Z}_q^* \\
\quad\quad\quad Y_i^a = (1/r_i^*)X - (s_i^*/r_i^*)P \\
\quad\quad\quad \text{If } Y_i^a = \mathcal{O}, \text{ goto } \dagger \\
\quad \text{For } i = 1, \ldots, n: \\
\quad\quad \ddagger\ t_i \leftarrow \mathbb{Z}_q^* \\
\quad\quad\quad Y_i^b = Y + t_i P \\
\quad\quad\quad \text{If } Y_i^b = \mathcal{O}, \text{ goto } \ddagger \\
\quad (m_0, m_1, A, B, state) \leftarrow A_f^{Sim}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b) \\
\quad d \stackrel{r}{\leftarrow} \{0,1\} \\
\quad \kappa_1^* || \kappa_2^* \stackrel{r}{\leftarrow} \{0,1\}^{l_e} \times \{0,1\}^{l_h} \\
\quad c^* \leftarrow \mathcal{E}_{\kappa_1^*}(m_d) \\
\quad str^* \leftarrow X || m_d || \kappa_2^* || Y_A^a || Y_B^b \\
\quad \text{Add } (str^*, r_A^*) \text{ to } L^{H_2} \\
\quad \sigma^* \leftarrow (c^*, r_A^*, s_A^*) \\
\quad \alpha^* \leftarrow e(Y_B^b, X) \\
\quad d' \leftarrow A_g^{Sim}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b, \sigma^*, m_0, m_1, state)
\end{array}
$$

Theorem 1. *Let $SC = (\mathcal{SP}, \mathcal{K}, \mathcal{S}, \mathcal{U}, \mathcal{N}, \mathcal{PV})$ be as in Definition 4, using hash functions H_1 and H_2, and symmetric encryption scheme $\mathcal{SE}$. In the random oracle model for H_1 and H_2 we have*

$$
\begin{aligned}
\mathbf{Adv}_{SC}^{ind-cca}(1^k, t, n) \leq\ & 4 \cdot \mathbf{Adv}_{\mathcal{SP}}^{cdh}(1^k, t') + \mathbf{Adv}_{\mathcal{SE}}^{ind-cpa}(1^k, t', 0) + \frac{q_{h_2}}{2^{l_h - 1}} \\
& + \frac{2(q_{h_2} + q_s + 3q_u + 3q_n)}{q-1} + \frac{(q_s + n)(2q_{h_2} + 2q_s + 2q_u + 2q_n + 5)}{(q-1)} \qquad (2)
\end{aligned}
$$

Here, if E is the complexity of computing e and M is the complexity of a multiplication in $\langle P \rangle$ (where P is returned by $\mathcal{SP}(1^k)$), then $t' = t + O(nM) + O(q_h E) + O(q_s(E + M)) + O((q_u + q_n)(E + M))$.

Proof. Suppose that $A_{cca} = (A_f, A_g)$ is an adversary that defeats the IND-CCA security of SC in the n-user setting as in Definition 5. Let A_{cca} be such that it runs for time at most t, makes at most q_s signcryption queries, q_u unsigncryption queries, q_n non-repudiation queries, q_{h_1} queries to H_1, and q_{h_2} queries to H_2.

To begin with $\mathcal{SP}(1^k)$ is run to produce group information I. Suppose that we are given $X = xP$ and $Y = yP$. We will construct algorithms to show that A_{cca}'s advantage is bounded by:

(1) The advantage in solving CDH problem.
(2) The advantage in breaking $\mathcal{SE}$.

Consider the algorithm in Figure 1 to solve the CDH problem in $\langle P \rangle$. The meanings of Sim and L^{H_2} are explained below.

Since we are in the random oracle model, we must provide subroutines to simulate responses to the H_1 and H_2 queries made by A_{cca}. We must also provide subroutines to

FIGURE 2

Algorithm $H_{1sim}(U)$	Algorithm $H_{2sim}(str)$
$\alpha \leftarrow e(U,P)$	If $(str, r) \in L^{H_2}$ for some r:
* If $\alpha = \alpha^*$:	Return r
$Z \leftarrow U - t_B X$	$r \xleftarrow{r} \mathbb{Z}_q^*$
RETURN Z	Add (str, r) to L^{H_2}
If $(-, \kappa_1 \| \kappa_2, \alpha) \in L_2^{H_1}$ for some $\kappa_1 \| \kappa_2$:	Return r
Return $\kappa_1 \| \kappa_2$	
If $(U, \kappa_1 \| \kappa_2, \alpha) \in L_1^{H_1}$ for some $\kappa_1 \| \kappa_2$:	
Return $\kappa_1 \| \kappa_2$	
$\kappa_1 \| \kappa_2 \xleftarrow{r} \{0,1\}^{l_e} \times \{0,1\}^{l_h}$	
Add $(U, \kappa_1 \| \kappa_2, \alpha)$ to $L_1^{H_1}$	
Return $\kappa_1 \| \kappa_2$	

simulate the signcryption oracle, the unsigncryption oracle and the non-repudiation oracle. The superscript Sim on A_f and A_g denotes this collection of simulators and the fact that $A_{cca} = (A_f, A_g)$ has access to them. Note that the if B_{cdh} is successful then the solution Z to the CDH problem is actually returned by one of its subroutines. See Figure 2 and Figure 5.

Let us first consider the simulation for H_1. We must keep a list L^{H_1} to maintain consistency between calls made by A_{cca}. We divide L^{H_1} into two parts $L^{H_1} = L_1^{H_1} \cup L_2^{H_1}$.

An entry of $L_1^{H_1}$ is of the form $(U, \kappa_1||\kappa_2, \alpha)$ where $(U, \kappa_1||\kappa_2)$ is the query/response pair and $\alpha = e(U,P)$ is stored for use later in the simulation. Note that the inputs to H_1 are bit strings and the inputs to the pairing are elliptic curve points. We are therefore making an implicit conversion from bit strings to elliptic curve points. Furthermore we are assuming that the adversary only queries H_1 with bit strings that represent elliptic curve points. This is justified since, in the random oracle model, querying H_1 with any other bit strings will be of no use to the adversary. Alternatively, we could have maintained two lists: one as above for strings that represent valid elliptic curve points; and one of simple query/response pairs for strings that do not represent valid elliptic curve points. The alternative solution would not change the result.

An entry of $L_2^{H_1}$ is of the form $(-, \kappa_1||\kappa_2, \alpha)$ where $\alpha = e(Y_z, V)$ for some Y_z and V. This represents the relation $H_1(x_z V) = \kappa_1||\kappa_2$ where $x_z = \mathrm{DLog}_P(Y_z)$ which we do not know.

The H_2 simulation and book keeping is much simpler. We keep a list L^{H_2}, an entry of which consists of simple query/response pair (str, r).

The lists L^{H_1} and L^{H_2} are available to all algorithms, as are $I, X, Y, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b$. The H_1 and H_2 simulations are described in Figure 2. The step marked with $*$ is only executed when responding to a query from A_g. Note that the step "RETURN Z" in $H_{1sim}(U)$ is the step that returns the Diffie-Hellman value for algorithm B_{cdh}.

The signcryption and unsigncryption oracle simulators are described in Figure 3, Figure 4 and Figure 5. Note that the algorithm of Figure 5 is a subroutine that is only called by the algorithm of Figure 4. Also, the step "RETURN Z" of $\mathcal{U}'_{\mathcal{O}sim}$ in Figure 5 is a step that returns the Diffie-Hellman value for algorithm B_{cdh}. The unsigncryption simulation will

FIGURE 3

Algorithm $\mathcal{S}_{\mathcal{O}sim}(Y_i^a, Y_j^b, m)$
$\dagger\ r, s \xleftarrow{r} \mathbb{Z}_q^*$
$V \leftarrow sP + rY_i^a$
If $V = \mathcal{O}$, goto $\dagger$
$\alpha \leftarrow e(Y_j^b, V)$
If $(U, \kappa_1'||\kappa_2', \alpha) \in L_1^{H_1}$ for some $(U, \kappa_1'||\kappa_2')$:
$\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
Else if $(-, \kappa_1'||\kappa_2', \alpha) \in L_2^{H_1}$ for some $\kappa_1'||\kappa_2'$:
$\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
Else:
$\kappa_1||\kappa_2 \leftarrow \{0,1\}^{l_e} \times \{0,1\}^{l_h}$
Add $(-, \kappa_1||\kappa_2, \alpha)$ to $L_2^{H_1}$
$c \leftarrow \mathcal{E}_{\kappa_1}(m)$
$str \leftarrow V||m||\kappa_2||Y_i^a||Y_j^b$
Add (str, r) to L^{H_2}
$\sigma \leftarrow (c, r, s)$
Return σ

differ depending whether it is responding to a query from A_f or A_g. The lines marked with $*$ are only executed when responding to a query from A_g. Algorithm $\mathcal{N}_{\mathcal{O}sim}$ is constructed in the same way as $\mathcal{U}_{\mathcal{O}sim}$. However, where $\mathcal{U}_{\mathcal{O}sim}$ returns m, $\mathcal{N}_{\mathcal{O}sim}$ returns Y_i^a, Y_j^b, m and (κ_2, r, s).

Let us now examine our simulation. We consider how A_{cca} runs in a real attack (*real*) and in the above simulation (*sim*). Define an event BAD to be one that causes the joint distribution of A_{cca}'s view to differ in *sim* from the distribution of A_{cca}'s view in *real*.

$$\begin{aligned}
\Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD}]_{sim} &= \Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD}]_{real} \\
&\geq \Pr[A_{cca} \text{ wins}]_{real} - \Pr[\mathsf{BAD}]_{real} \\
&= \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{ind-cca}_{SC,A_{cca}}(1^k) - \Pr[\mathsf{BAD}]_{real}. \qquad (3)
\end{aligned}$$

We define the event $\mathtt{FIND}$ to be that where A_g makes a H_1 query U such that $e(U,P) = \alpha^*$. We have

$$\begin{aligned}
&\Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD}]_{sim} \\
&= \Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD} \wedge \mathtt{FIND}]_{sim} + \Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD}\neg \wedge \neg\mathtt{FIND}]_{sim} \\
&\leq \Pr[\mathtt{FIND}]_{sim} + \Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD}\neg \wedge \neg\mathtt{FIND}]_{sim}. \qquad (4)
\end{aligned}$$

Now, if this $\mathtt{FIND}$ occurs in sim then B_{cdh} succeeds in solving the CDH problem which means that (4) becomes

$$\begin{aligned}
&\Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD}]_{sim} \\
&\leq \mathbf{Adv}^{cdh}_{\mathcal{SP},B_{cdh}}(1^k) + \Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD}\neg \wedge \neg\mathtt{FIND}]_{sim}. \qquad (5)
\end{aligned}$$

FIGURE 4

Algorithm $\mathcal{U}_{\mathcal{O}sim}(Y_i^a, Y_j^b, \sigma)$
  Parse σ as (c, r, s)
  If $r \notin \mathbb{Z}_q^* \vee s \notin \mathbb{Z}_q^*$, return $\perp$
  $V \leftarrow sP + rY_i^a$
  If $V = \mathcal{O}$, return $\perp$
  $\alpha \leftarrow e(Y_j^b, V)$
  If $(U, \kappa_1'||\kappa_2', \alpha) \in L_1^{H_1}$ for some $(U, \kappa_1'||\kappa_2')$:
    $\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
* Else if $\alpha = \alpha^*$, call $\mathcal{U}'_{\mathcal{O}sim}(Y_i^a, Y_j^b, \sigma)$
  Else if $(-, \kappa_1'||\kappa_2', \alpha) \in L_2^{H_1}$ for some $\kappa_1'||\kappa_2'$
    $\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
  Else:
    $\kappa_1||\kappa_2 \xleftarrow{r} \{0,1\}^{l_e} \times \{0,1\}^{l_h}$
    Add $(-, \kappa_1||\kappa_2, \alpha)$ to $L_2^{H_1}$
  $m \leftarrow \mathcal{D}_{\kappa_1}(c)$
  $str \leftarrow V||m||\kappa_2||Y_i^a||Y_j^b$
  If $(str, r') \in L^{H_2}$ for some r':
    If $r' = r$:
      Return m
    Return $\perp$
  $r' \xleftarrow{r} \mathbb{Z}_q^*$
  Add (str, r') to L^{H_2}
  If $r' = r$, return m
  Return $\perp$

We have from (3) and (5) that

$$\begin{aligned} &\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{ind-cca}_{SC,A_{cca}}(1^k) - \Pr[\mathsf{BAD}]_{real} \\ (6) \qquad &\leq \mathbf{Adv}^{cdh}_{\mathcal{SP},B_{cdh}}(1^k) + \Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD} \wedge \neg\mathtt{FIND}]_{sim}. \end{aligned}$$

We next establish a bound on $\Pr[\mathsf{BAD}]_{real}$.

The only way in which A_{cca}'s view can differ in *real* and *sim* is if there is an error in B_{cdh}, $H_{1\,sim}$, $H_{2\,sim}$, $\mathcal{S}_{\mathcal{O}sim}$ or $\mathcal{U}_{\mathcal{O}sim}$. We therefore split

$$(7) \qquad \mathsf{BAD} = \mathsf{BAD}_{B_{cdh}} \vee \mathsf{BAD}_{H_1} \vee \mathsf{BAD}_{H_2} \vee \mathsf{BAD}_{\mathcal{S}_{\mathcal{O}}} \vee \mathsf{BAD}_{\mathcal{U}_{\mathcal{O}}}$$

The only possible error in B_{cdh} is trying to insert $(X||m_d||\kappa_2^*||Y_A^a||Y_B^b, r_A^*)$ into L^{H_2} when $(X||m_d||\kappa_2^*||Y_A^a||Y_B^b, r) \in L^{H_2}$ for $r \neq r_A^*$. At the point when this could happen X is outside A_{cca}'s view. Therefore, for such an error, X would have to occur by chance in a query made by A_{cca} or in a response to such a query. We infer

$$(8) \qquad \Pr[\mathsf{BAD}_{B_{cdh}}]_{real} \leq \frac{q_{h_2} + q_s + q_u + q_n}{q-1}.$$

It is easy to see that $H_{1\,sim}$ and $H_{2\,sim}$ provide perfect simulations and therefore

FIGURE 5

```
Algorithm U'_{O sim}(Y_i^a, Y_j^b, σ)
    Parse σ as (c, r, s)
    If r ∉ Z_q^* ∨ s ∉ Z_q^*:
    If Y_j^b = Y_B^b:
        If r = r_i^*, return ⊥
        Else:
            x ← (r_i^* s + r s_i^*)/(r_i^* − r)
            Z ← xY
            RETURN Z
    Else if Y_i^a = Y_A^a and r = r_i^*, return ⊥
    Else if r = r_i^* ABORT
    Else:
        β ← (r_i^* s t_j − r s_i^* t_j)/(r_i^* − r) mod q
        γ ← (r t_j − r_i^* t_B)/(r_i^* − r) mod q
        δ ← (r_i^* s − r s_i^*)/(r_i^* − r) mod q
        Z ← βP + γX + δY
        RETURN Z
```

$$\Pr[\mathsf{BAD}_{H_1}]_{real} = \Pr[\mathsf{BAD}_{H_2}]_{real} = 0. \tag{9}$$

Let us now consider $\Pr[\mathsf{BAD}_{\mathcal{S}_{\mathcal{O}}} \wedge \neg\mathsf{BAD}_{B_{cdh}}]$. We define U^* to be such that $e(U^*, P) = e(Y_B^b, X) = \alpha^*$, where B is the target receiver chosen by A_f. There are three possible cases described below for an error in $\mathcal{S}_{\mathcal{O}\,sim}$.

(1) An error could be caused by $\mathcal{S}_{\mathcal{O}\,sim}$ in the find stage if, at the end of this stage, A_f outputs B such that an invocation of $\mathcal{S}_{\mathcal{O}\,sim}$ has added $(-, \kappa_1||\kappa_2, e(Y_j^b, V))$ to $L_2^{H_1}$ such that $e(Y_j^b, V) = e(Y_B^b, X)$. Since X is independent of A_f's view this can happen with probability at most $q_s/(q-1)$.

(2) An error could be caused by $\mathcal{S}_{\mathcal{O}\,sim}$ in the guess stage if, the query U^* has never been made to H_1, and the simulator, in responding to a query (Y_i^a, Y_j^b, m), generates r and s such that $e(Y_j^b, rP + sY_i^a) = e(Y_B^b, X)$. This can occur with probability at most $q_s/(q-1)$.

(3) The other possibility of an error caused by $\mathcal{S}_{\mathcal{O}\,sim}$ can occur in either stage. This error is cased if $\mathcal{S}_{\mathcal{O}\,sim}$ produces r and str such that $(str, r') \in L^{H_2}$ for $r' \neq r$, or $str = str^*$. This can occur with probability at most

$$\sum_{i=0}^{q_s-1} \frac{q_{h_2} + q_u + q_n + 1 + i}{q-1} = \frac{q_s(2q_{h_2} + q_s + 2q_u + 2q_n + 1)}{2(q-1)}.$$

The three cases above tell us

$$\Pr[\mathsf{BAD}_{\mathcal{S}_{\mathcal{O}}} \wedge \neg\mathsf{BAD}_{B_{cdh}}]_{real} \leq \frac{q_s(2q_{h_2} + q_s + 2q_u + 2q_n + 5)}{2(q-1)}. \tag{10}$$

We now deal with $\Pr[\mathsf{BAD}_{\mathcal{U}_\mathcal{O}} \wedge \neg\mathsf{BAD}_{\mathcal{S}_\mathcal{O}} \wedge \neg\mathsf{BAD}_{B_{cdh}}]_{real}$. We will split this up into two cases: possible errors in the find stage and possible errors in the guess stage.

An error can only be caused by $\mathcal{U}_{\mathcal{O}sim}$ in the find stage if A_f makes a query (Y_i^a, Y_j^b, σ), where $\sigma = (c, r, s)$, such that the public key Y_B^b corresponding to user B output by A_f satisfies $e(Y_B^b, X) = e(Y_j^b, rP + sY_i^a)$. Since X is independent of A_f's view we infer

(11) $$\Pr[\mathsf{BAD}_{\mathcal{U}_\mathcal{O}}(\text{find}) \wedge \neg\mathsf{BAD}_{\mathcal{S}_\mathcal{O}} \wedge \neg\mathsf{BAD}_{B_{cdh}}]_{real} \leq \frac{q_u + q_n}{q-1}.$$

It remains to analyse

(12) $$\Pr[\mathsf{BAD}_{\mathcal{U}_\mathcal{O}}(\text{guess}) \wedge \neg\mathsf{BAD}_{\mathcal{U}_\mathcal{O}}(\text{find}) \wedge \neg\mathsf{BAD}_{\mathcal{S}_\mathcal{O}} \wedge \neg\mathsf{BAD}_{B_{cdh}}]_{real}$$

Let us denote the event $\neg\mathsf{BAD}_{\mathcal{U}_\mathcal{O}}(\text{find}) \wedge \neg\mathsf{BAD}_{\mathcal{S}_\mathcal{O}} \wedge \neg\mathsf{BAD}_{B_{cdh}}$ by $\mathsf{3NB}$ (3 Not Bad). So, from (12), we are interested in

(13) $$\Pr[\mathsf{BAD}_{\mathcal{U}_\mathcal{O}}(\text{guess}) \wedge \mathsf{3NB}]_{real}.$$

In the guess stage the only possibility for a $\mathcal{U}_{sim}$ error is rejecting a valid ciphertext. We know that if this occurs then A_g must have made a $\mathcal{U}_{\mathcal{O}sim}$ query with $(Y_i^a, Y_j^b, (c, r, s))$ such that

$$e(Y_j^b, rP + sY_i^a) = \alpha^*.$$

To analyse the probability of this occurring we must look at $\mathcal{U}'_{\mathcal{O}sim}$ in Figure 5 since this deals with all such queries. We split this into three cases.

`CASE1`. Event that would cause $\mathcal{U}'_{\mathcal{O}sim}$ to (incorrectly) return $\perp$
`CASE2`. Event that would cause $\mathcal{U}'_{\mathcal{O}sim}$ to abort
`CASE3`. Event that would cause $\mathcal{U}'_{\mathcal{O}sim}$ to return Z on behalf of B_{cdh}

Let us suppose that $\mathcal{U}'_{\mathcal{O}sim}$ (incorrectly) returns $\perp$ in response to some query $Y_i^a, Y_j^b, \sigma = (c, r, s)$. In this case the following conditions are satisfied.

C1. $e(Y_j^b, rP + sY_i^a) = \alpha^*$
C2. $r = H_2(X||m||\kappa_2^*||Y_i^a||Y_j^b)$ where $m \leftarrow \mathcal{D}_{\kappa_1^*}(c)$
C3. $X||m||\kappa_2^*||Y_i^a||Y_j^b \neq X||m_d||\kappa_2^*||Y_A^a||Y_B^b$

The conditions C1 and C2 are obvious. To see why C3 must hold we use an argument adapted from the full version of [4].

Suppose that C1 and C2 both hold and that $X||m||\kappa_2^*||Y_i^a||Y_j^b = X||m_d||\kappa_2^*||Y_A^a||Y_B^b$. This tells us that the following conditions are satisfied.

D1. $e(Y_j^b, rP + sY_i^a) = e(Y_B^b, X)$ and since $Y_j^b = Y_B^b$, $rP + sY_A^a = X$
D2. $c = c^*$ (since $m = \mathcal{D}_{\kappa_1^*}(c) = \mathcal{D}_{\kappa_1^*}(c^*) = m_d$ and the fact that $\mathcal{D}$ is injective for any key)
D3. $r = r_A^*$ (since otherwise $\mathcal{U}'_{\mathcal{O}sim}$ does not return $\perp$)
D3. $s = s^*$ (by the properties of $\langle P \rangle$ and the facts that $sP + rY_A^a = s_A^*P + r_A^*Y_A^a$)

We infer that if we had $X||m||\kappa_2^*||Y_i^a||Y_j^b = X||m_d||\kappa_2^*||Y_A^a||Y_B^b$ would mean that (c, r, s) was the target ciphertext which is not allowed. Therefore the only way $\mathcal{U}'_{\mathcal{O}sim}$ returns $\perp$ incorrectly is if H_2 outputs one of $\{r_1^*, \ldots, r_n^*\}$ in response to some query. Over all queries made to $\mathcal{U}_{\mathcal{O}sim}$ and $\mathcal{N}_{\mathcal{O}sim}$ we infer that

(14) $$\Pr[\mathtt{CASE1}]_{real} \leq \frac{n(q_{h_2} + q_s + q_u + q_n)}{q-1}.$$

Note that in the above we have assumed that $X||m||\kappa_2^*||Y_i^a||Y_j^b = X||m_d||\kappa_2^*||Y_A^a||Y_B^b$ means that $Y_i^a = Y_A^a$ and $Y_j^b = Y_B^b$ as elliptic curve points. We can ensure that this is the case by insisting that the bit string representation of all points in $\langle P \rangle$ are of fixed length.

The simulator $\mathcal{U}'_{\mathcal{O}\,sim}$ only aborts if $r = r_i^*$ and $Y_i^a \neq Y_A^a$. When $Y_i^* \neq Y_A^a$, r_i^* is unknown to the adversary and so

$$\Pr[\texttt{CASE2}]_{real} \leq \frac{q_u + q_n}{q-1}. \tag{15}$$

We must now analyse

$$\begin{aligned} &\Pr[\texttt{CASE3} \wedge \neg\texttt{CASE2} \wedge \neg\texttt{CASE1} \wedge \texttt{3NB}]_{real} \\ =&\Pr[\texttt{CASE3} \wedge \neg\texttt{CASE2} \wedge \neg\texttt{CASE1} \wedge \texttt{3NB}]_{sim} \end{aligned} \tag{16}$$

The equality in (16) concerning the equality of probabilities in $real$ and sim follows from a fact that we have been implicitly using in our analysis of $\Pr[\texttt{BAD}]_{real}$: all the events so far (other than CASE3) depend on collisions being caused by the output of random number generators or random oracles; such events occur with equal probability in $real$ and in sim. Moreover, this is true until CASE3 occurs.

It is verified in the Appendix that if CASE3 occurs, $\mathcal{U}_{\mathcal{O}\,sim}$ returns Z which is the correct solution to the Diffie-Hellman problem. From this, (14), (15) and (16) it is clear that

$$\begin{aligned} &\Pr[\texttt{BAD}_{\mathcal{U}_\mathcal{O}}(\text{guess}) \wedge \texttt{3NB}]_{real} \\ \leq&\frac{n(q_{h_2}+q_s+q_u+q_n)}{q-1} + \frac{q_u+q_n}{q-1} + \mathbf{Adv}^{cdh}_{\mathcal{SP},B_{cdh}}(1^k) \end{aligned} \tag{17}$$

Now, from (7), (8), (9), (10), (11), (12), (13), (17) we have

$$\begin{aligned} \Pr[\texttt{BAD}]_{real} \leq \frac{q_{h_2}+q_s+3q_u+3q_n}{q-1} &+ \frac{(q_s+n)(2q_{h_2}+2q_s+2q_u+2q_n+5)}{2(q-1)} \\ &+ \mathbf{Adv}^{cdh}_{\mathcal{SP},B_{cdh}}(1^k). \end{aligned} \tag{18}$$

The final stage is to give a bound on $\Pr[A_{cca} \text{ wins} \wedge \neg\texttt{BAD} \wedge \neg\texttt{FIND}]_{sim}$. Let ASK be the event in which A_{cca} makes the H_2 query $X||m_d||\kappa_2^*||Y_A^a||Y_B^b$. We have

$$\begin{aligned} &\Pr[A_{cca} \text{ wins} \wedge \neg\texttt{BAD} \wedge \neg\texttt{FIND}]_{sim} \\ =&\Pr[A_{cca} \text{ wins} \wedge \neg\texttt{BAD} \wedge \neg\texttt{FIND} \wedge (\texttt{ASK} \vee \neg\texttt{ASK})]_{sim} \\ \leq&\frac{q_{h_2}}{2^{l_h}} + \Pr[A_{cca} \text{ wins} \wedge \neg\texttt{BAD} \wedge \neg\texttt{FIND} \wedge \neg\texttt{ASK}]_{sim}. \end{aligned} \tag{19}$$

This follows from the fact that in the event $\neg$FIND, κ_2^* is unknown to A_{cca}.

To complete the proof we show how, in the event A_{cca} wins $\wedge\ \neg$BAD $\wedge\ \neg$FIND $\wedge$ $\neg$ASK, it would be possible to use A_{cca} to break the symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_{se}, \mathcal{E}, \mathcal{D})$ in the sense of Definition 8. We describe an adversary A_{se} to do this in Figure 6. The collection of subroutines Sim necessary for the execution of A_{se} are those used by B_{cdh} of Figure 1.

In the event A_{cca} wins $\wedge\ \neg$BAD $\wedge\ \neg$FIND $\wedge\ \neg$ASK, the following are true of A_{se} in Figure 6.

- The adversary A_{se} requires no access to an $\mathcal{E}$ oracle.
- Adversary A_{cca} is run by A_{se} in exactly the same way as it would be run by B_{cdh}.
- If A_{cca} wins, then A_{se} also wins.
- The time taken to run A_{se} is the same as that taken to run B_{cdh}.

FIGURE 6

Algorithm $A_{se_1}(1^k)$
For $i = 1, \ldots, n$:
† $r_i^*, s_i^* \leftarrow \mathbb{Z}_q^*$
$Y_i^a = (1/r_i^*)X - (s_i^*/r_i^*)P$
If $Y_i^a = \mathcal{O}$, goto †
For $i = 1, \ldots, n$:
‡ $t_i \leftarrow \mathbb{Z}_q^*$
$Y_i^r = Y + t_i P$
If $Y_i^r = \mathcal{O}$, goto ‡
$(m_0, m_1, A, B, state) \leftarrow A_f^{Sim}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
$state' \leftarrow (state, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b, r_A^*, s_A^*)$
Return $(m_0, m_1, state')$

Outside of A_{se}'s view κ_1^* is chosen at random from $\{0,1\}^{l_e}$ and a bit d is chosen uniformly at random. Message m_d is encrypted under κ_1^* to produce $c^* \leftarrow \mathcal{S}_{\kappa_1^*}(m_d)$.

Algorithm $A_{se_2}(1^k, c^*, m_0, m_1, state')$
Pares $state'$ as $(state, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b, r_A^*, s_A^*)$
$\sigma^* \leftarrow (c^*, r_A^*, s_A^*)$
$d' \leftarrow A_g^{Sim}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b, \sigma^*, m_0, m_1, state)$
Return d'

From the above observations and examining the construction of A_{se} we infer that

$$\Pr[A_{cca} \text{ wins} \wedge \neg\mathsf{BAD} \wedge \neg\mathsf{FIND} \wedge \neg\mathsf{ASK}]_{sim} \leq \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{ind-cpa}_{\mathcal{SE},A_{se}}(1^k)$$

$$\leq \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{ind-cpa}_{\mathcal{SE}}(1^k, t_2, 0). \tag{20}$$

with $t_2 = t + O(nM) + O(q_{h_1}E) + O(q_s(E+M)) + O((q_u + q_n)(E+M))$, where E is the complexity of evaluating e and M is the complexity of a multiplication in $\langle P \rangle$.

The result now follows from (6), (18), (19) and (20). □

4.2. **Unforgeability of the scheme SC.** In this section we will prove SC to be secure under Definition 6. In our security analysis we will assume that, for a security parameter 1^k, $\mathcal{SP}(1^k)$ has been run to produce I. Subsequently $\mathcal{K}(I)$ is run to produce (x, Y) and we are given Y.

We will show how an adversary, in the sense of Definition 6, may be used to recover x from Y. Let A_1 be such an adversary that attacks SC in an n-user setting. In its attack A_1 makes at most q_{h_1}, q_{h_2}, q_s, q_u and q_n queries to H_1, H_2, the signcryption oracle, the unsigncryption oracle and the non-repudiation oracle respectively.

We will run A_1 in a simulated environment to construct an adversary A_2 that makes zero queries to the signcryption oracle, zero queries to the random oracle H_1, zero queries to the unsigncryption oracle, zero queries to the non-repudiation oracle and one query to the random oracle H_2. To make this possible we describe a collection of simulators H_{1sim}, H_{2sim}, $\mathcal{S}_{\mathcal{O}sim}$, $\mathcal{U}_{\mathcal{O}sim}$ and $\mathcal{N}_{\mathcal{O}sim}$ using the same notation as Theorem 1. We denote this collection of simulators Sim. We keep lists L^{H_1} and L^{H_2} that are initially empty to maintain consistency between calls to H_{1sim} and H_{2sim}. These lists have the same form as those in Theorem 1.

First of all we generate the n pairs of public keys to supply to our adversary with as in Figure 7. Once we have done this we are ready to run A_2. We describe A_2 in Figure 8. We describe the simulations with which it interacts in Figure 9, Figure 10 and Figure 11. Algorithm $\mathcal{N}_{\mathcal{O}sim}$ is constructed in the same way as $\mathcal{U}_{\mathcal{O}sim}$. Where $\mathcal{U}_{\mathcal{O}sim}$ returns m, $\mathcal{U}_{\mathcal{O}sim}$ returns Y_i^a, Y_j^b, m and (κ_2, r, s).

FIGURE 7

Sender keys	**Receiver keys**
$x_1^a \leftarrow 1$	For $i = 1, \ldots, n$:
$Y_1^a \leftarrow Y$	$\quad x_i^b \xleftarrow{r} \mathbb{Z}_q^*$
For $i = 2, \ldots, n$:	$\quad Y_i^b \leftarrow x_i^b P$
$\quad x_i^a \xleftarrow{r} \mathbb{Z}_q^*$	
$\quad Y_i^a \leftarrow x_i^a Y$	

FIGURE 8

Algorithm $A_2^{H_2}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
$J \xleftarrow{r} \{1, \ldots, q_h\}$
$Y_i^a, Y_B, m^*, (\kappa_2^*, r^*, s^*) \leftarrow A_1^{Sim}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
If $r^* \neq \tilde{r}$:
ABORT
Return $Y_i^a, Y_B, m^*, (\kappa_2^*, r^*, s^*)$

FIGURE 9

Algorithm $H_{1sim}(U)$
$\alpha \leftarrow e(U, P)$
If $(-, \kappa_1||\kappa_2, \alpha) \in L_2^{H_1}$ for some $\kappa_1||\kappa_2$:
Return $\kappa_1||\kappa_2$
If $(U, \kappa_1||\kappa_2, \alpha) \in L_1^{H_1}$ for some $\kappa_1||\kappa_2$:
Return $\kappa_1||\kappa_2$
$\kappa_1||\kappa_2 \xleftarrow{r} \{0,1\}^{l_e} \times \{0,1\}^{l_h}$
Add $(U, \kappa_1||\kappa_2, \alpha)$ to $L_1^{H_1}$
Return $\kappa_1||\kappa_2$

Algorithm $H_{2sim}(string)$
If this is the J-th query to H_{2sim}:
$\tilde{r} \leftarrow H_2(string)$
If $(string, r') \in L^{H_2}$ for $r' \neq \tilde{r}$:
ABORT
Add $(string, \tilde{r})$ to L^{H_2}
Return $\tilde{r}$
If $(string, r) \in L^{H_2}$ for some r:
Return r
$r \xleftarrow{r} \mathbb{Z}_q^*$
Add $(string, r)$ to L^{H_2}
Return r

FIGURE 10

Algorithm $\mathcal{S}_{\mathcal{O}sim}(Y_i^a, Y_j^b, m)$
$\dagger$ $r, s \xleftarrow{r} \mathbb{Z}_q^*$
$V \leftarrow sP + rY_i^a$
If $V = \mathcal{O}$ goto $\dagger$
$\alpha \leftarrow e(Y_j^b, V)$
If $(U, \kappa_1'||\kappa_2', \alpha) \in L_1^{H_1}$ for some $U, \kappa_1'||\kappa_2'$:
$\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
Else if $(-, \kappa_1'||\kappa_2', \alpha) \in L_2^{H_1}$ for some $\kappa_1'||\kappa_2'$:
$\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
Else:
$\kappa_1||\kappa_2 \xleftarrow{r} \{0,1\}^{l_e} \times \{0,1\}^{l_h}$
Add $(-, \kappa_1||\kappa_2, \alpha)$ to $L_2^{H_1}$
$c \leftarrow \mathcal{E}_{\kappa_1}(m)$
$string \leftarrow V||m||\kappa_2||Y_i^a||Y_j^b$
If $(string, r') \in L^{H_2}$ for $r' \neq r$, ABORT
Return (c, r, s)

FIGURE 11

Algorithm $\mathcal{U}_{\mathcal{O}sim}(Y_i^a, Y_j^b, \sigma)$
Parse σ as (c, r, s)
If $r \notin \mathbb{Z}_q^* \vee s \notin \mathbb{Z}_q^*$:
Return $\perp$
$V \leftarrow sP + rY_i^a$
If $V = \mathcal{O}$, return $\perp$
$\alpha \leftarrow e(Y_j^b, V)$
If $(U, \kappa_1'||\kappa_2', \alpha) \in L_1^{H_1}$ for some $U, \kappa_1'||\kappa_2'$:
$\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
Else if $(-, \kappa_1'||\kappa_2', \alpha) \in {L_2}^{H_1}$ for some $\kappa_1'||\kappa_2'$:
$\kappa_1||\kappa_2 \leftarrow \kappa_1'||\kappa_2'$
Else:
$\kappa_1||\kappa_2 \xleftarrow{r} \{0,1\}^{l_e} \times \{0,1\}^{l_h}$
Add $(-, \kappa_1||\kappa_2, \alpha)$ to ${L_2}^{H_1}$
$m \leftarrow \mathcal{D}_{\kappa_1}(c)$
$string \leftarrow V||m||\kappa_2||Y_i^a||Y_j^b$
If $(string, r'') \in L^{H_2}$ for some r'', $r' \leftarrow r''$
Else:
$r' \leftarrow \mathbb{Z}_q^*$
Add $(string, r')$ to L^{H_2}
If $r' = r$, return m
Else, return $\perp$

Claim 1.

$$\mathbf{Adv}^{cma}_{SC,A_2}(1^k) \geq \frac{1}{q_{h_2}}\left(\mathbf{Adv}^{cma}_{SC,A_1}(1^k) - \frac{q_{h_2}(q_{h_2}+2(q_s+q_u)-1)+2}{2(q-1)}\right) \tag{21}$$

Proof. Suppose that A_1 succeeds in forging $Y_i^a, Y_B, m^*, (\kappa_2^*, r^*, s^*)$. Let $V^* \leftarrow s^*P + r^*Y_i^a$ and let $string^* \leftarrow V^*||m^*||\kappa_2^*||Y_i^a||Y_B$. We define some special events below.

- Event CQ: Adversary A_1 makes the query $string^*$ to the random oracle H_2 during its successful attack. The initials stand for critical query.
- Event Q_i: For $i = 1, \ldots, q_{h_2}$, Q_i is the event that A_1 makes the query $string^*$ to the random oracle H_2 for the first time at the i-th query.

Let us now consider the advantage of A_2. We will denote probabilities in A_1's real attack, i.e. the experiment given in Definition 6, with a subscript *real* and denote probabilities where A_1 is run by A_2 with a subscript *sim*. From the construction of A_2 it is clear that

$$\begin{aligned}
\mathbf{Adv}^{cma}_{SC,A_2}(1^k) &\geq \sum_{i=1}^{q_{h_2}} \Pr[(J=i) \wedge A_1 \text{ wins} \wedge Q_i]_{sim} \\
&= \sum_{i=1}^{q_{h_2}} \Pr[J=i]_{sim} \Pr[A_1 \text{ wins} \wedge Q_i]_{sim} \\
&= \frac{1}{q_{h_2}} \Pr[A_1 \text{ wins} \wedge CQ]_{sim} \\
&\geq \frac{1}{q_{h_2}}\left(\Pr[A_1 \text{ wins} \wedge CQ]_{real} - \frac{q_{h_2}(q_{h_2}+2(q_s+q_u)-1)}{2(q-1)}\right) \qquad (22) \\
&= \frac{1}{q_{h_2}}\Bigg(\Pr[A_1 \text{ wins}]_{real} - \Pr[A_1 \text{ wins} \wedge \neg CQ]_{real} \\
&\qquad\qquad - \frac{q_{h_2}(q_{h_2}+2(q_s+q_u)-1)}{2(q-1)}\Bigg) \\
&\geq \frac{1}{q_{h_2}}\left(\mathbf{Adv}^{cma}_{SC,A_1}(1^k) - \frac{q_{h_2}(q_{h_2}+2(q_s+q_u)-1)+2}{2(q-1)}\right) \qquad (23)
\end{aligned}$$

For (22) note that A_1 can only win in the simulation if $\mathcal{S}_{\mathcal{O}\,sim}$ does not abort. The probability of $\mathcal{S}_{sim}$ aborting is at most

$$\frac{q_{h_2}(q_{h_2}+2(q_s+q_u)-1)+2}{2(q-1)}.$$

For (23) note that, in the random oracle model, A_1 can not win with probability greater than $\frac{1}{q-1}$ if the event CQ does not occur.

There is an implicit assumption above that the query Q_J of A_1 in its successful run is not for a string that has been added to L^{H_2} by an invocation of $\mathcal{S}_{\mathcal{O}\,sim}$. That is to say we assume that $H_{2\,sim}$ does not abort. This is acceptable because a successful run of A_1 must output a message that was never a signcryption query during the attack. □

The computational assumption on which our security proof relies will be the discrete logarithm assumption. We define the particular instance of the problem that we will be interested in below.

FIGURE 12

Algorithm $A_3(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
$r_0, r_1 \xleftarrow{r} \mathbb{Z}_q^*$
If $r_0 = r_1$ ABORT
Start $A_2(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
If the subroutine $H_{2\,sim}$ calls H_2 respond with r_0
$Y_i^a, Y_B, m^*, (\kappa_2^*, r_0^*, s_0^*) \leftarrow A_2^{H_2}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
Start $A_2(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$ a second time with the same random input
If the subroutine $H_{2\,sim}$ calls H_2 respond with r_1
$Y_i^a, Y_B, m^*, (\kappa_2^*, r_1^*, s_1^*) \leftarrow A_2^{H_2}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$
$x \leftarrow (s_0^* - s_1^*)/\,(x_i^a(r_1^* - r_0^*)) \mod q$
Return x

Definition 9 (The discrete logarithm assumption)**.** *Let $\mathcal{SP}$ and $\mathcal{K}$ be as in Definition 4 of SC. Let A be an algorithm and consider the experiment below.*

Experiment $\mathbf{Exp}^{dlog}_{\mathcal{SP},A}(1^k)$
$I \xleftarrow{r} \mathcal{SP}(1^k)$
$(x, Y) \xleftarrow{r} \mathcal{K}(I)$
$x \leftarrow A(1^k, I, Y)$
If $Y = xP$ return 1, else return 0

We define the advantage of the algorithm A as

$$\mathbf{Adv}^{dlog}_{\mathcal{SP},A}(1^k) = \Pr[\mathbf{Exp}^{dlog}_{\mathcal{SP},A}(1^k) = 1].$$

For any integers t, we define the advantage function *as*

$$\mathbf{Adv}^{dlog}_{\mathcal{SP}}(1^k, t) = \max_A\{\mathbf{Adv}^{dlog}_{\mathcal{SP},A}(1^k)\},$$

where the maximum is over all algorithms with time complexity t.

The discrete logarithm assumption is that $\mathbf{Adv}^{dlog}_{\mathcal{SP}}(1^k, t)$ is a negligible function of the security parameter.

From A_2 we now construct an algorithm A_3 to find the discrete logarithm of Y. When A_2 is run by A_3 it no longer has access to H_2. We describe A_3 in Figure 12.

Let us suppose that we have a boolean matrix where each row corresponds to a possible random input for the adversary $A_2^{H_2}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$ and each column corresponds to a possible random response to the H_2 query made by A_2. We denote this matrix $\Phi(RA_2, r)$. An entry of $\Phi(RA_2, r)$ is one if $A_2^{H_2}(1^k, I, Y_1^a, Y_1^b, \ldots, Y_n^a, Y_n^b)$ wins when run with the appropriate random input and H_2 response, and zero otherwise.

Definition 10 (Heavy row)**.** *Let $\Phi(RA_2, r)$ be as above. A heavy row of $\Phi(RA_2, r)$ is one where the fraction of ones is at least $\frac{1}{2}\mathbf{Adv}^{cma}_{SC,A_2}(n, k)$.*

Our proof will require the simple lemma below from [28].

Lemma 1 (Heavy row lemma)**.** *If $\mathbf{Adv}^{cma}_{SC,A_2}(n, k) \geq \frac{4}{q-1}$ then the ones of $\Phi(RA_2, r)$ are located in heavy rows with probability $\frac{1}{2}$.*

Proof. For compactness we will denote $\Phi(RA_2, r)$ by Φ and $\mathbf{Adv}^{cma}_{SC,A_2}(n,k)$ by $\mathbf{Adv}_{A_2}$.

$$\begin{aligned}
&\Pr[\text{ In heavy row of } \Phi \mid \text{Entry of } \Phi \text{ is } 1\,] \\
=&1-\Pr[\,\neg \text{ In heavy row of } \Phi \mid \text{Entry of } \Phi \text{ is } 1\,] \\
=&1-\frac{\Pr[\,\neg \text{ In heavy row of } \Phi\ \wedge\ \text{Entry of } \Phi \text{ is } 1\,]}{\Pr[\text{ Entry of } \Phi \text{ is } 1\,]} \\
\geq&1-\frac{\frac{1}{2}\mathbf{Adv}_{A_2}}{\mathbf{Adv}_{A_2}}=\frac{1}{2}
\end{aligned}$$

□

Claim 2. *If* $\mathbf{Adv}^{cma}_{SC,A_2}(n,k) \geq \frac{4}{q-1}$ *then*

(24) $$\mathbf{Adv}^{dlog}_{\mathcal{SP},A_3}(1^k) \geq \frac{1}{4}\Big(\mathbf{Adv}^{cma}_{SC,A_2}(n,k)\Big)^2 - \frac{1}{q-1}.$$

Proof. Consider an execution of A_3 with random input ρ. Assuming that $r_0 \neq r_1$, the first call to A_2 made by A_3 succeeds with probability $\mathbf{Adv}^{cma}_{SC,A_2}(n,k)$. If the first call to A_2 is successful, by Lemma 1, the entry of the matrix $\Phi(RA_2, r)$ corresponding to ρ and r_0 is in a heavy row with probability $\frac{1}{2}$. If we are in a heavy row, with probability $\frac{1}{2}\mathbf{Adv}^{cma}_{SC,A_2}(n,k)$ the entry of $\Phi(RA_2, r)$ corresponding to ρ and r_1 is 1. This gives us the probability that the second call to A_2 made by A_3 is successful. It is easily verified that when $r_0 \neq r_1$ and both calls to A_2 made by A_3 are successful then A_3 is successful. The result follows. □

From (21), (24) and the construction of A_2 and A_3 we derive the result below.

Theorem 2. *Let* $SC = (\mathcal{SP}, \mathcal{K}, \mathcal{S}, \mathcal{U}, \mathcal{N}, \mathcal{PV})$ *be as in Definition 4, using hash functions* H_1 *and* H_2*, and symmetric encryption scheme* $\mathcal{SE}$*. In the random oracle model for* H_1 *and* H_2 *we have*

$$\begin{aligned}
\mathbf{Adv}^{cma}_{SC}(1^k, t, q_s, q_u, q_{h_1}, q_{h_2}) \leq 2q_h\Big(\mathbf{Adv}^{dlog}_{\mathcal{SP}}(1^k, t_1) + \frac{1}{q-1}\Big)^{\frac{1}{2}} \\
+\frac{q_{h_2}(q_{h_2}+2(q_s+q_u)-1)}{2(q-1)}
\end{aligned}$$

with $t_1 = t + O(nM) + O(q_{h_1}E) + O(q_s(E+M)) + O((q_u+q_n)(E+M))$*, where* E *is the complexity of evaluating* e *and* M *is the complexity of a multiplication in* $\langle P\rangle$*.*

Note that, since we are in the random oracle model, the advantage $\mathbf{Adv}^{cma}_{SC}(\ k, t, q_s, q_u$ $,q_{h_1},q_{h_2})$ is now the maximum over all adversaries running for time t, making at most q_{h_1}/q_{h_2} queries to H_1/H_2, at most q_s signcryption queries and at most q_u unsigncryption queries.

5. Conclusion

We have described a model of security for signcryption schemes that offer non-interactive non-repudiation. This is a model of security in the multi-user setting. Our model applies to existing schemes such as those in [5, 23, 30].

We have shown how the scheme of [5] is insecure under any definition of privacy based on indistinguishability of signcryptions. We have proposed a modification of this scheme to overcome the weakness. Proofs of security of the modified scheme have been given in the random oracle model. We note that this scheme is almost as efficient as the scheme in [4], the only significant extra cost is one group exponentiation in the signcryption operation and one in unsigncryption.

REFERENCES

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer-Verlag, 2001.

[2] J. H. An. Authenticated encryption in the public-key setting: Security notions and analyses, 2001. Available at http://eprint.iacr.org/2001/079/.

[3] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer-Verlag, 2002.

[4] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. In *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer-Verlag, 2002.

[5] F. Bao and R. H. Deng. A signcryption scheme with signature directly verifiable by public key. In *Public Key Cryptography '98*, volume 1431 of *Lecture Notes in Computer Science*, pages 55–59. Springer-Verlag, 1998.

[6] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for paring-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.

[7] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer-Verlag, 2000.

[8] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In 38^{th} *Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Science Press, 1997.

[9] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 1998.

[10] M. Bellare, M. Jakobsson, and M. Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 280–305. Springer-Verlag, 1997.

[11] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In 1^{st} *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[12] M. Bellare and P. Rogaway. Optimal asymmetric encryption - how to encrypt with RSA. In *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.

[13] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001. Full version available at http://eprint.iacr.org/2001/090/.

[14] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer System Sciences*, 37:156–189, 1988.

[15] J. C. Cha and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Public Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, 2003.

[16] D. Chaum and T. P. Pederson. Wallet databases with observers. In *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.

[17] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.

[18] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Available at http://eprint.iacr.org/2001/108/, 2001.

[19] S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory (ANTS V)*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.

[20] S. Galbraith, J. Malone-Lee, and N. P. Smart. Public key signatures in the multi-user setting. *Information Processing Letters*, 83(5):263–266, 2002.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

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

[23] W. H. He and T. C. Wu. Cryptanalysis and improvement of Petersen-Michels signcryption scheme. *IEE Proceedings - Computers and Digital Techniques*, 146(2):123–124, 1999.

[24] F. Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer-Verlag, 2003.

[25] M. K. Lee, D. K. Kim, and K. Park. An authenticated encryption scheme with public verifiability. In 4^{th} *Korea-Japan Joint Workshop on Algorithms and Computation*, pages 49–56, 2000.

[26] J. Malone-Lee. Identity-based signcryption, 2002. Available at http://eprint.iacr.org/2002/098/.

[27] A. J. Menezes, T. Okamato, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[28] K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369. Springer-Verlag, 1998.

[29] K. G. Patterson. ID-based signatures from pairings on elliptic curves. *Electronic Letters*, 38(18):1025–1026, 2002.

[30] H. Petersen and M. Michels. Cryptanalysis and improvement of signcryption schemes. *IEE Proceedings - Computers and Digital Techniques*, 145(2):149–151, 1998.

[31] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 235–251. Springer-Verlag, 1990.

[32] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[33] N. P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronic Letters*, 38(13):630–632, 2002.

[34] E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer-Verlag, 2001.

[35] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) $<<$ cost(signature) $+$ cost(encryption). In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 1997.

APPENDIX

Suppose that $\mathcal{U}'_{\mathcal{O}sim}$ or $\mathcal{N}'_{\mathcal{O}sim}$ is responding to a query (Y_i^a, Y_j^b, σ) made by A_g where $\sigma = (c, r, s)$. Let

$$V = sP + rY_i^a. \tag{25}$$

Recall that this query is such that $e(Y_j^b, V) = e(Y_B^b, X) = \alpha^*$. This would mean that

$$\mathrm{DLog}_P(Y_j^b) \cdot \mathrm{DLog}_P(V) = \mathrm{DLog}_P(Y_B^b) \cdot \mathrm{DLog}_P(X) \mod q. \tag{26}$$

We split the event of such a query into two cases: `CASE1` is the event that $Y_j^b = Y_B^b$ in the offending query; `CASE2` is the event that $Y_j^b \neq Y_B^b$ in the offending query.

`CASE1`: $Y_j^b = Y_B^b$
In this instance we must have

$$x = \mathrm{DLog}_P(X) = \mathrm{DLog}_P(V) \tag{27}$$

From (25) and the way that Y_i^a is created in Figure 1, we infer that

$$\mathrm{DLog}_P(V) = s + r\frac{1}{r_i^*}x - r\frac{s_i^*}{r_i^*} \tag{28}$$

Assuming $r_i^* \neq r$, together (27) and (28) give us

(29) $$\mathrm{DLog}_P(X) = x = \frac{r_i^* s + r s_i^*}{r_i^* - r}$$

Using (29) we can compute

$$xyP = xY$$

as required.

CASE2: $Y_j^b \neq Y_B^b$
Let us denote

(30) $$x = \mathrm{DLog}_P(X) \text{ and } y = \mathrm{DLog}_P(Y)$$

From the way that Y_i^a, Y_j^b and Y_B^b are created in Figure 1, we infer that

$$\mathrm{DLog}_P(Y_j^b) = y + t_j P$$
$$\mathrm{DLog}_P(Y_B^b) = y + t_B P$$
$$\mathrm{DLog}_P(V) = s + r\left(\frac{x}{r_i^*} - \frac{s_i^*}{r_i^*}\right)$$

These equations and (26) tell us that

(31) $$(y + t_B)x = (y + t_j)\left(s + r\left(\frac{x}{r_i^*} - \frac{s_i^*}{r_i^*}\right)\right)$$

Assuming $r_i^* \neq r_i$, define

$$\beta = \frac{r_i^* s t_j - r s_i^* t_j}{r_i^* - r}$$
$$\gamma = \frac{r t_j - r_i^* t_B}{r_i^* - r}$$
$$\delta = \frac{r_i^* s - r s_i^*}{r_i^* - r}$$

By equating coefficients in (31) and using the definitions for β, γ and δ above we have

$$xyP = \beta P + \gamma X + \delta Y$$

as required.

E-mail address: malone@cs.bris.ac.uk

UNIVERSITY OF BRISTOL, DEPARTMENT OF COMPUTER SCIENCE, MERCHANT VENTURERS BUILDING, WOODLAND ROAD, BRISTOL, BS8 1UB, UK