Signcryption with Non-interactive Non-repudiation without Random Oracles

Jia Fan[1,2], Yuliang Zheng[1], and Xiaohu Tang[2]

[1] University of North Carolina at Charlotte, NC 28223, USA
[2] Southwest Jiaotong University, 610031, P.R. China
{jfan1,yzheng}@uncc.edu, xhutang@ieee.org

Abstract. Non-repudiation is a very important requirement of signcryption. It ensures that a sender cannot deny the fact that he has signcrypted a message. Non-interactive non-repudiation enables a receiver to settle a repudiation dispute with the help of a judge without the need to engage in costly multi-round interactive communications with the judge. In this paper, we strengthen Malone-Lee's security model for signcryption with non-interactive non-repudiation by introducing two additional, more subtle and useful security requirements, one about the unforgeability and the other about the confidentiality of non-repudiation evidence. A further contribution of this paper is to design a concrete signcryption scheme that admits provable security without random oracles in our strengthened security model for signcryption.

Keywords: signcryption, non-repudiation, public key cryptography, non-interaction, random oracle, bilinear map.

1 Introduction

Asymmetric encryption and signature are two basic primitives in public-key cryptography. They provide us with confidentiality and authenticity independently. When both functions are required, traditionally one has to carefully sign and encrypt the data sequentially. In 1997, Zheng [26] proposed a new primitive called signcryption. It combines the functions of both primitives with a cost much less than the sign-then-encrypt (or encrypt-then-sign) method.

Let us consider a scenario where a sender signcrypts a message which is then forwarded to a receiver. Afterwards the sender denies the fact. We note that in the original signcryption, only the receiver can decrypt the signcryptext, that is, he is the only one who can check the validity of the message. The challenge the receiver faces is what he can do to ask a judge to help prove the fact, while without revealing to the judge more information than that is required. Non-repudiation is defined to guarantee that the sender cannot deny the fact that the message is signcrypted by her in the first place.

One technique suggested by Zheng [26] is to rely on a judge who can be totally trusted. In this case, a receiver simply gives his private key to the judge. The judge can decrypt the signcryptext and verify the validity of the message by

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. X, LNCS 6340, pp. 202–230, 2010.

making use of the receiver's private key. A second technique suggested by Zheng deals with a situation where the judge is not fully trusted. With the second method, the receiver engages an interactive zero-knowledge proof protocol with the judge. At the end of the execution of the protocol, the judge can make a decision as to whether the signcrytext is indeed from the sender. Clearly, the second method suggested by Zheng is not quite efficient in practice.

Bao and Deng [3] proposed a modified signcryption scheme, with the aim of offering non-repudiation in a non-interactive way. With their method, when there is a dispute on a message M and a signcryptext σ between a receiver R and a sender S, the receiver R computes some non-repudiation evidence d, and forwards (M, σ, d) together with the public keys (PK_S, PK_R) to a not necessarily trusted judge. The judge can verify whether S has signcrypted M into σ for the receiver R. However, it was later pointed out in [15] and [20] that the non-repudiation evidence d would destroy the confidentiality of the message.

To address problems with Bao and Deng's scheme, Malone-Lee [15] proposed a new security model specifically for signcryption with non-interactive non-repudiation (NINR). This model ensures that the exposure of evidence d does not ruin the security of both confidentiality and unforgeability.

Our model. Now a natural question to ask is whether a signcryption scheme in Malone-Lee's model can be assured to be provably secure. We will show that the answer to the question is unfortunately negative. The main reason for this is that Malone-Lee's model addresses only two basic security requirements, namely confidentiality and unforgeability, which turns out to be inadequate to properly define the model of signcryption with NINR. We now analyze it in greater detail.

First, it is required that a given piece of evidence d can help the judge make a correct decision, especially when a given M is not the unsigncryption result of a given σ. It turns out that Malone-Lee's model does not provide this guarantee. As an example we examine a signcryption scheme proposed by Chow et al [9]. Interestingly, although that scheme can be proved to be secure in Malone-Lee's model, a piece of not well-formed evidence d can lead a judge to incorrectly regard a wrong message M' as being the unsigncryption of a signcryptext σ. To rectify the above problem, we consider a new security requirement for signcryption with NINR, namely soundness of non-repudiation. Fulfilling this requirement will guarantee that a judge can always make a right decision.

Second, we observe that in some previous schemes, such as those proposed in [17] [18] [14] [24], non-repudiation evidence d can be generated not only by the receiver but also by the sender. That is to say, even if a judge is sure that a signcryptext σ is in fact signcrypted from some message M, he still can not be sure who generated this non-repudiation evidence d. This ambiguity can cause troubles in many practical uses. As an example, consider a patient who receives a signcrypted medical report from his doctor. If the patient is malicious, he can generate a piece of well-formed evidence d, and then deliberately expose the contents of the report to a third party. Latter, he claims that it is the doctor who exposes his report to the third party, and asks for compensation. A judge in this case will not be able to decide who, the patient or the doctor, is on the

wrong side. Problems of similar nature may occur in many other situations, e.g. military scenarios, on-line business transactions etc. In order to clarify the above ambiguity, we consider an additional new security requirement, namely unforgeability of non-repudiation evidence, which guarantees that only the receiver can generate valid non-repudiation evidence d.

Our scheme. Since the concept of signcryption introduced by Zheng [26], a number of signcryption schemes with the property of non-interactive non-repudiation [15] [17] [18] [9] etc. have been designed and proved secure in the random oracle model [6] which assumes that certain functions, such as one-way hash functions, output truly random values. While the random oracle model has been a very useful tool in the field of provable security, no real hash function behaves like a random function. As a result, designing a signcryption scheme with NINR that does not rely on a random oracle for its security is both attractive in scholarly research and useful in practice. In the past few years, a number of research papers e.g.[22] [23] [14] have been published on the topic of signcryption without random oracles. However, according to the best of our knowledge, none of these schemes is provably secure for non-interactive non-repudiation.

In this paper, we design a signcryption scheme with NINR that can be proved secure without random oracles. Our signcryption scheme is based on the signature scheme of Boneh, Shen and Waters [8], and is very compact when compared with the underling signature scheme. We will provide a specific efficiency comparison in Section 5.1.

Organization. The rest of the paper is organized as follows: We introduce some preliminary facts in Section 2. In Section 3 we describe our model for signcryption with NINR by defining the syntax, analyzing Malone-Lee's model, defining four security requirements, together with in depth discussions on core aspects of the model. In Section 4, we construct a concrete signcryption scheme with NINR, and prove that it is secure without random oracles. In Section 5, we discuss how to improve the efficiency of the scheme together with its practical applications. Finally, we draw some conclusions in Section 6. As a side contribution, we note that our contruction can be turned into an even more efficient scheme when random oracles are allowed. We discuss this in the appendix.

2 Preliminaries

2.1 Bilinear Maps

Throughout this paper we use the following standard notations on bilinear maps.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. A symmetric bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: all $u, v \in \mathbb{G}$ satisfy $e(u, v) \neq 1$.

2.2 Collision Resistent Hash Functions

Throughout this paper we use ϵ with an appropriate subscript to indicate a negligible function that vanishes at least as fast as the inverse of a polynomial in an appropriate security parameter.

A hash function H is said to be collision resistant if it is infeasible for an adversary to find two different inputs m_0 and m_1 such that $H(m_0) = H(m_1)$. A more formal definition follows.

Definition 1. *A hash function H is (t, ϵ_H)-collision-resistant if for any adversary $\mathcal{A}$ running in time t, it has possibility at most ϵ_H in finding two different inputs m_0 and m_1 such that $H(m_0) = H(m_1)$.*

We require two collision resistent functions with different ranges for their outputs. Specifically, let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of prime order p. The first collision resistent function H_1 maps input from $\mathbb{G}_T \times \mathbb{G} \times \mathbb{G}$ to an element in $\mathbb{Z}_p$, and the second resistent function H_2 maps input from $\mathbb{G}$ to a string in $\{0,1\}^n$.

2.3 Discrete Logarithm Assumption

The discrete logarithm problem applies to mathematical structures called groups. Let $\mathbb{G}$ be a group of prime order p, and g be a generater for $\mathbb{G}$. We have the following definition for the discrete logarithm (D-Log) assumption.

Definition 2. *The (t, ϵ_{DLog}) D-Log assumption holds in $\mathbb{G}$, if for any adversary $\mathcal{A}$, given a random element $g_3 \in \mathbb{G}$, running in time t, $\mathcal{A}$ has possibility at most ϵ_{DLog} in finding an integer $x \in Z_p$ such that $g^x = g_3$.*

2.4 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Let $\mathbb{G}$, $\mathbb{G}_T$ be groups of a same prime order p, g be a generater of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Choose a, b, c, k from $\mathbb{Z}_p$ at random, and let

$$BDH = \{g, g^a, g^b, g^c, T \leftarrow e(g,g)^{abc}\},$$
$$Random = \{g, g^a, g^b, g^c, T \leftarrow e(g,g)^{k}\}.$$

The DBDH assumption claims that BDH and $Random$ are indistinguishable. For any adversary $\mathcal{A}$, consider two experiments. $\mathcal{A}$ is given BDH in experiment 0, and is given $Random$ in experiment 1. $\mathcal{A}$'s advantage for solving the DBDH assumption is

$$\epsilon_{dbdh} = |Pr[\mathcal{A} = 1 \text{ in experiment } 0] - Pr[\mathcal{A} = 1 \text{ in experiment } 1]|.$$

Definition 3. *The (t, ϵ_{dbdh})-DBDH assumption holds, if any adversary $\mathcal{A}$ running in time t has advantage at most ϵ_{dbdh} in solving the DBDH assumption.*

3 The Proposed Model of Signcryption with NINR

3.1 Syntax of Signcryption with NINR

A signcryption scheme with NINR is composed of six algorithms. The first four algorithms constitute a signcryption scheme, and the last two algorithms fulfill the requirements of NINR.

- SetupPub(1^η), run by a trusted party: Given a security parameter 1^η, a trusted party generates and outputs the system's public parameter Pub.
- KeyGen(Pub, ID_P), run by every user: User P takes the public parameter Pub as input, outputs a pair of private/public keys (SK_P, PK_P).
- Signcryption(SK_S, PK_R, M), run by a sender: To communicate a message $M \in \mathcal{M}$ ($\mathcal{M}$ is the message space) from a sender S to a receiver R, the algorithm produces a signcryptext σ on M by using S's private key SK_S and R's public key PK_R. The signcryptext σ is sent to R.
- Unsigncryption(SK_R, PK_S, σ), run by a receiver: Upon receiving a signcryptext σ from S, the algorithm first checks whether σ is valid. It returns a plaintext M if σ is valid, or a special symbol $\perp$ otherwise.
- NR-Evidence-Gen(SK_R, PK_S, σ), run by a receiver: If σ is a valid signcryptext, the algorithm computes and returns a piece of non-repudiation evidence d. Otherwise, the algorithm returns a symbol $\perp$.
- JG-Verification $(\sigma, M, d, PK_S, PK_R)$, run by a judge: Upon receiving a signcryptext/message pair (σ, M), a piece of non-repudiation evidence d, a sender S' public key PK, and a receiver R's public key PK_R, the algorithm returns a special symbol $\top$ if it is S who has signcrypted the message M into σ for R, or a symbol $\perp$ otherwise.

For consistency, we require that for all $\sigma = Signcryption(SK_S, PK_R, M)$, we should have $M = Unsigncryption(SK_R, PK_S, \sigma)$.

For completeness, we require that for all signcryptext σ and all possible $d = NR\text{-}Evidence\text{-}Gen(SK_R, PK_S, \sigma)$, if $M = Unsigncryption(SK_R, PK_S, \sigma)$, then we should have $\top \leftarrow JG\text{-}Verification(\sigma, d, M, PK_S, PK_R)$.

Remark 1. *The public parameter Pub is not explicitly taken as input to the last four algorithms, since we assume that all the users in the system know Pub.*

3.2 Analysis of Malone-Lee's Model

We first review security models for regular signcryption. Baek *et al.* [8] proposed a formal security model for signcryption in 2001. Independently of this, An *et al.* [1] also came up with similar security models for signcryption. Both models consider two security definitions, namely confidentiality and unforgeability. And in the models of both papers, two factors are considered:

1. If there are only two users (a sender and a receiver) in the network, then it is called a two-user setting; otherwise if there are many (more than two) users in the network, then it is called a multi-user setting.

2. If the adversary is a sender (in the attack game of unforgeability) or a receiver (in the attack game of confidentiality) in the communication for challenge, then we call it an inside attacker setting. Otherwise, we call it an outside attacker setting.

Melone-Lee's model [15] is different from the widely used definitions proposed by Baek *et al.* [8] and An *et al.*[1] in the following two aspects:

1. In Molone-Lee's model, an adversary $\mathcal{A}$ is able to get the value of evidence d by asking for non-repudiation oracles. For each non-repudiation oracle, $\mathcal{A}$ makes queries with a signcryptext σ together with a sender and a receiver's public keys, and receives as a return from the oracle a piece of corresponding evidence d.

2. Malone-Lee's model is defined in a multi-user attacker setting, but the basic underling security definitions are different from the definitions proposed by Baek *et al.* [8] and An *et al.*[1]. For example, in the attack game (for either confidentiality or unforgeability) with a multi-user inside attacker setting in [1] and [8], the adversary is able to generate public keys for all users in the system except the one who is an attack target. In comparison, with the attack game (e.g. confidentiality) of Malone-Lee's model, the adversary is given public keys for all users in the system at the beginning. Afterwards, he chooses one of them as his attack target. An advantage of Malone-Lee's model is that the user (whom the adversary will attack against) can be arbitrarily chosen by the attacker. But the total number of all users in the system needs to be pre-decided, and the public keys of all users should be pre-computed by the simulator. When there are a large number of users in the system (which happens frequently in practise), security bounds provided by the proof become less tight.

3.3 Security Definitions in Our Model

In our model, we will consider four security requirements, namely confidentiality, unforgeability, soundness of non-repudiation, and unforgeability of non-repudiation evidence. If a signcryption scheme with NINR can be proved secure under the first three definitions, we say that it is *SCNINR secure*. If a signcryption scheme with NINR is SCNINR secure and can also be proved secure under the definition of unforgeability of non-repudiation evidence, we say that it is *strong SCNINR secure*.

Our definitions do not follow Malone-Lee's model directly. Instead we mainly refer to the basic definitions of [8] and [1] in a multi-user inside attacker setting, together with Malone-Lee's idea [15] of adding non-repudiation oracles in the attack game.

Confidentiality. The attack game for indistinguishability of signcryption under chosen ciphertext attack (IND-SCNINR-CCA) contains five steps as follows:

– *Setup system:* An adversary $\mathcal{A}$ is given the system's public parameter $Pub \leftarrow SetupPub(1^\eta)$, and a challenge user B's public key $PK_B \leftarrow KeyGen(Pub, ID_B)$.

- *Oracles before challenge:* $\mathcal{A}$ is able to ask for a number of signcryption, unsigncryption and non-repudiation oracle queries associated with the challenge user B.
  - For each signcryption oracle query, $\mathcal{A}$ generates a receiver's public key PK_R, a message $M \in \mathcal{M}$, and outputs (PK_S, PK_R, M) with $PK_S = PK_B$. This oracle returns to $\mathcal{A}$ with $\sigma \leftarrow Signcryption(SK_B, PK_R, M)$.
  - For each unsigncryption oracle query, $\mathcal{A}$ generates a sender's public key PK_S and a signcryptext σ, outputs (PK_S, PK_R, σ) with $PK_R = PK_B$. This oracle returns to $\mathcal{A}$ with the result of $Unsigncryption(PK_S, SK_B, \sigma)$.
  - For each non-repudiation oracle query, $\mathcal{A}$ generates a sender's public key PK_S, a signcryptext σ, and outputs (PK_S, PK_R, σ) with $PK_R = PK_B$. This oracle returns to $\mathcal{A}$ with the result of $NR\text{-}Evidence\text{-}Gen(PK_S, SK_B, \sigma)$.
- *Challenge:* $\mathcal{A}$ generates a sender's public key PK_{S^*}, and produces two equal length messages (M_0, M_1) in $\mathcal{M}$. $\mathcal{A}$ outputs $(PK_{S^*}, PK_{R^*}, M_0, M_1)$ with $PK_{R^*} = PK_B$, then is returned with $\sigma^* \leftarrow Signcryption(SK_{S^*}, PK_B, M_\gamma)$, where γ is randomly chosen from $\{0, 1\}$.
- *Oracles after challenge:* This step is the same as Oracles before challenge step, except that $\mathcal{A}$ is not allowed to ask for an unsigncrypiton oracle query or a non-repudiation oracle query on σ^* with sender/receiver public key $(PK_{S^*}, PK_{R^*} = PK_B)$.
- *Guess:* $\mathcal{A}$ outputs a guess bit γ' for γ.

If $\gamma' = \gamma$, then $\mathcal{A}$ wins the above attack game. We define the advantage for $\mathcal{A}$ to win this game is $\epsilon = |Pr[\gamma' = \gamma] - 1/2|$.

Definition 4. *The signcryption scheme with NINR is $(t, q_s, q_u, q_n, \epsilon)$ IND-SCNINR-CCA secure, if for running in time t, any adversary $\mathcal{A}$ who has asked for signcryption oracle queries q_s times, unsigncryption oracle queries q_u times and non-repudiation oracle queries q_n times, has advantage at most ϵ in winning the IND-SCNINR-CCA game.*

Unforgeability. The attack game for strong existential unforgeability of signcryption with NINR under chosen message attack (SEU-SCNINR-CMA) contains three steps as follows:

- *Setup system:* The same as the Setup system step in the IND-SCNINR-CCA game.
- *Oracles:* The same as the Oracles before the challenge step in the IND-SCNINR-CCA game.
- *Forge:* $\mathcal{A}$ generates a receiver's public key PK_{R^*}, and outputs a forged signcryptext σ^* on (PK_{S^*}, PK_{R^*}) with $PK_{S^*} = PK_B$.

If the following two conditions are both satisfied, then we say that $\mathcal{A}$ wins the SEU-SCNINR-CMA game:

1. $Unsigncryption(PK_B, SK_{R^*}, \sigma^*) \neq \perp$;
2. σ^* is not a result of any the signcryption oracle queries with sender/receiver public key $(PK_{S^*} = PK_B, PK_{R^*})$.

Definition 5. *The signcryption scheme with NINR is* $(t, q_s, q_u, q_n, \epsilon)$ *SEU-SCNINR-CMA secure, if for running in time* t*, any adversary* $\mathcal{A}$*, who has asked for signcryption oracle queries* q_s *times, unsigncryption oracle queries* q_u *times and non-repudiation oracle queries* q_n *times, has possibility at most* ϵ *in winning the SEU-SCNINR-CMA game.*

Soundness of Non-repudiation. As we have described in the introduction, soundness of non-repudiation should ensure a judge always make a right decision. That is, if a given M is not the unsigncryption result of a given σ, the judge should not let it pass the verification. We first give an intuition for the attack game:

To achieve this goal, our attack game described below for the soundness of non-repudiation assumes a very strong adversary $\mathcal{A}$, who can generate all users' public/private keys, including the challenge user B. Then in the challenge, $\mathcal{A}$ asks for one signcryption oracle. He outputs (M, PK_S), the signcryption oracle returns a signcryptext $\sigma = Signcryption(SK_S, PK_R = PK_B, M)$. Finally, if $\mathcal{A}$ outputs another message $M'(M' \neq M)$, and a piece of evidence d' such that $JG\text{-}Verification(\sigma, M', d', PK_S, PK_B) = \top$, then $\mathcal{A}$ wins.

In this attack game, we do not have oracle stages(as in pervious attack games), since $\mathcal{A}$ is stronger than the attackers (in confidentiality and unforgeability). $\mathcal{A}$ knows all users' public/private keys, therefore, he can compute all the algorithms in the scheme himself. Finally, if $\mathcal{A}$ wins, it implies the judge makes a wrong decision.

This definition is similar to the definition of proof soundness in the model of public-key encryption with non-interactive opening by Damgard *et al.* [11] and Galindo *et al.* [12].

The game for the soundness of non-repudiation of signcryption with NINR consists of three steps as follows:

- *Setup system:* First, the adversary $\mathcal{A}$ is given the system's public parameter Pub. Then he generates a challenge user B's public/private key pair (PK_B, SK_B), and forwards (PK_B, SK_B) to the system.
- *Challenge:* In this stage, $\mathcal{A}$ has access to a signcryption oracle query once. $\mathcal{A}$ generates a sender's public key PK_S and a message $M \in \mathcal{M}$, then outputs (PK_S, PK_R, M) with $PK_R = PK_B$ to the signcryption oracle. Finally, $\mathcal{A}$ is returned with $\sigma \leftarrow Signcryption(SK_S, PK_B, M)$.
- *Output:* $\mathcal{A}$ outputs a message M' together with some non-repudiation evidence d'.

If $JG\text{-}Verification(\sigma, M', d', PK_S, PK_B) = \top$ and $M' \neq M$, then $\mathcal{A}$ wins this game.

Definition 6. *A SCNINR scheme satisfies* (t, ϵ) *computational the soundness of non-repudiation, if any adversary running in time* t *has probability at most* ϵ *in winning the above game where* ϵ *is negligible. If* $\epsilon = 0$*, the SCNINR scheme satisfies the perfect soundness of non-repudiation.*

Unforgeability of Non-repudiation Evidence. The attack game is similar to the attack game of unforgeability in most stages, but is different in the forge stage. The adversary's object here is to forge a piece of valid non-repudiation evidence on a new signcryptext.

The game for existential unforgeability of non-repudiation evidence in signcryption with NINR under chosen message attack (EUF-NR-evidence-SCNINR-CMA) contains three steps as follows:

- *Setup system:* The same as the Setup system step in SEU-SCNINR-CMA game.
- *Oracles:* The same as the Oracles step in the SEU-SCNINR-CMA game.
- *Forge:* $\mathcal{A}$ generates the sender's public key PK_{S^*}, outputs a message M^*, a piece of non-repudiation evidence d^*, and a signcryptext σ^*.

$\mathcal{A}$ wins the game if $JG\text{-}Verification(\sigma^*, d^*, M^*, PK_{S^*}, PK_{R^*}) = \top$ with $PK_{R^*} = PK_B$ and $\mathcal{A}$ has never asked for a non-repudiation oracle query or an unsigncryption oracle query on σ^* with sender/receiver public key (PK_{S^*}, PK_B).

Definition 7. *The signcryption scheme with NINR is* $(t, q_s, q_u, q_n, \epsilon)$ *EUF-NR-evidence-SCNINR-CMA secure if for running in time* t, $\mathcal{A}$ *has asked for* q_s *signcryption oracle queries,* q_u *unsigncryption oracle queries,* q_n *non-repudiation oracle queries and has possibility at most* ϵ *in winning this game.*

3.4 Adapt Our Model to Existing Schemes

When we adapt our model to existing schemes, we find out that some schemes e.g. Malone-Lee's scheme in [15] and the second scheme of Chow *et al.* in [9] achieve the first three security requirements[1], but no schemes can fulfill the security requirement of unforgeability of non-repudiation evidence. The reason why none of the existing schemes (including [15] and [9]) meets the last security requirement is that, traditionally, the evidence d is secret information (normally the Diffie-Hellman key) embedded by the sender to prevent other users except for the receiver from verifying the regular signature of M. In this method, the sender is the one who directly generates d, and the receiver can regenerate the value of d indirectly. In other words, both the sender and the receiver hold d. In signcryption, this results in an attack on the security requirement of unforgeability of non-repudiation evidence (as the sender can be a successful forger). For example, the evidence in Malone-Lee's scheme [15] is an element k_2 which can be generated by the receiver as well as the sender.

Our construction, which will be described in detail in the next section, is different from the traditional idea. The generation of evidence d makes use of the identity-based technique [7]. If one takes the receiver's private key as a master key of the public key generator (PKG), then d can be regarded as a private key

[1] Since the proofs are long and can be readily derived from existing proofs of those schemes, we omit them from this paper.

of identity ID whose value is determined by the current signcryptext σ. The judge decrypts σ by making use of d, and then checks whether it matches the value of the given message M. Informally, since d can only be used to decrypt σ rather than other signcryptexts, the exposure of d does not pose risks to confidentiality. Furthermore only the receiver, who is the only one holding the master key, can generate d. Therefore, the unforgeability of non-repudiation evidence is also ensured.

4 The Proposed Signcryption Scheme with NINR

4.1 Construction

Our signcryption scheme with NINR follows the six algorithm approach we defined in Section 3.1. We first describe the SetupPub algorithm, and then list other algorithms in Table 1 and Table 2.

- ***SetupPub***(1^η) by Trusted Party:
  On input a security parameter 1^η, a trusted party runs the following steps:
  1. Set up $\{\mathbb{G}, \mathbb{G}_T, e, g\}$, where $\mathbb{G}$ and $\mathbb{G}_\mathbb{T}$ are groups of prime order p, $g \in \mathbb{G}$ is a generator, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map.
  2. Set up $\{g_1, g_2, g_3, u_0, U\}$: Choose random elements g_1, g_2, g_3, u_0 from $\mathbb{G}$, and a random n-length vector $U = (u_1, ..., u_n) \in \mathbb{G}^n$. For each i $(1 \le i \le n)$, u_i is a random element in $\mathbb{G}$.
  3. Set up two collision-resistent hash functions H_1 and H_2, where $H_1 : \mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_p$ and $H_2 : \mathbb{G} \to \{0,1\}^n$.

  The system's public parameter is: $Pub = \{\mathbb{G}, \mathbb{G}_T, e, g, g_1, g_2, g_3, u_0, U, H_1, H_2\}$.

For consistency, one can verify that

$$\sigma_0 / e(\sigma_1, g_1^{\alpha_R}) = M \cdot e(g_1, g_R)^t / e(g^t, g_1^{\alpha_R}) = M.$$

For completeness, we have

$$\begin{aligned}
\frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) \cdot e(d_3, g_S)} &= \frac{M \cdot e(g_1, g_R)^t \cdot e(g_2^{\alpha_S} (u_0 \prod_{i=1}^n u_i^{c_i})^t, g^r)}{e(g^t, g_1^{\alpha_R} \cdot (u_0 \prod_{i=1}^n u_i^{c_i})^r) \cdot e(g_2^r, g_S)} \\
&= \frac{M \cdot e(g_1, g_R)^t \cdot e(g_2^{\alpha_S}, g^r) \cdot e(u_0 \prod_{i=1}^n u_i^{c_i}, g)^{t \cdot r}}{e(g_1, g_R)^t \cdot e(g, u_0 \prod_{i=1}^n u_i^{c_i})^{t \cdot r} \cdot e(g_2^r, g_S)} \\
&= M.
\end{aligned}$$

4.2 Security Proofs

Now we will prove that the above signcryption scheme with NINR is strong SCNINR secure.

Table 1. KeyGen& Signcryption & Unsigncryption Algorithms

KeyGen(Pub, ID_P) by User P: 1. randomly chooses $\alpha_P \in \mathbb{Z}_p$, 2. compute $g_P \leftarrow g^{\alpha_P}$, 3. let the private key be $SK_P \leftarrow \{\alpha_P\}$, 4. let the public key be $PK_P \leftarrow \{g_P\}$.
Signcryption(SK_S, PK_R, M) by Sender S: To signcrypt $M \in \mathbb{G}_T$ to be communicated to receiver R, sender S runs: 1. choose random elements $t, s \in \mathbb{Z}_p$, 2. compute $\sigma_0 \leftarrow M \cdot e(g_1, g_R)^t$, 3. compute $\sigma_1 \leftarrow g^t$, 4. compute $\theta \leftarrow H_1(\sigma_0, \sigma_1, g_S)$, 5. compute $z \leftarrow g^{\theta} {g_3}^s$, 6. compute $C \leftarrow H_2(z)$, write as $(c_1...c_n) \in \{0,1\}^n$, 7. compute $\sigma_2 \leftarrow g_2^{\alpha_S}(u_0 \prod_{i=1}^{n} u_i^{c_i})^t$, 8. set $\sigma_3 \leftarrow s$, 9. let the signcryptext be $\sigma \leftarrow (\sigma_0, \sigma_1, \sigma_2, \sigma_3)$.
Unsigncryption(SK_R, PK_S, σ) by Receiver R: To unsigncrypt σ from sender S, receiver R runs: 1. compute $\theta \leftarrow H_1(\sigma_0, \sigma_1, g_S)$, 2. compute $z \leftarrow g^{\theta} {g_3}^{\sigma_3}$, 3. compute $C \leftarrow H_2(z)$, and write it as $(c_1...c_n) \in \{0,1\}^n$, 4. if $e(\sigma_2, g) \neq e(g_2, g_S) \cdot e(\sigma_1, u_0 \prod_{i=1}^{n} u_i^{c_i})$, return $\perp$. 5. otherwise compute and return $M \leftarrow \sigma_0 / e(\sigma_1, g_1^{\alpha_R})$.

Proof of Confidentiality

Theorem 1. *The signcryption scheme is* $(t, q_s, q_u, q_n, \epsilon_{H_1} + \epsilon_{H_2} + \epsilon_{Dlog} + (q_u + q_n)/p + \epsilon_{dbdh})$ *IND-SCNINR-CCA secure, under the* (t, ϵ_{dbdh}) *DBDH assumption, the* (t, ϵ_{Dlog}) *Discrete Logarithm assumption in* $\mathbb{G}$*, and the assumption that the hash functions* H_1 *and* H_2 *are* (t, ϵ_{H_1}) *and* (t, ϵ_{H_2}) *collision resistent respectively.*

Proof of Theorem 1: We are going to use the game technique [19] to prove this theorem. Throughout this proof, we will list six games, from Game 0 to Game 5. All the games are executed between an adversary and a simulator. Game 0 is the IND-SCNINR-CCA game defined above, and other games will be quite similar to Game 0 in their overall structure, and will only differ from Game 0 in terms of how the simulator works. The key point for the proof is that we want to make sure that for each i $(1 \leq i \leq 5)$, either $Pr[\gamma = \gamma' \textit{ in game } i] = Pr[\gamma = \gamma' \textit{ in game } i-1]$ or $|Pr[\gamma = \gamma' \textit{ in game } i] - Pr[\gamma = \gamma' \textit{ in game } i-1]| \leq Pr[F_i]$ where $Pr[F_i]$ is a negligible value.

In order to analyze the value of $|Pr[\gamma = \gamma' \textit{ in game } i] - Pr[\gamma = \gamma' \textit{ in game } i-1]|$, we need the following lemma whose proof can be found in [19]:

Table 2. NR-Evidence-Gen & JG-Verification Algorithms

NR-Evidence-Gen(SK_R, PK_S, σ) by Receiver R: To compute non-repudiation evidence d, receiver R runs: 1. steps 1-4 of Unsigncryption in Table 1, 2. choose a random $r \in \mathbb{Z}_p$, 3. compute $d_1 \leftarrow g_1^{\alpha_R}(u_0 \prod_{i=1}^{n} u_i^{c_i})^r$, 4. compute $d_2 \leftarrow g^r$, 5. compute $d_3 \leftarrow g_2^r$, 6. return $d \leftarrow (d_1, d_2, d_3)$.
JG-Verification$(\sigma, M, d, PK_S, PK_R)$ by Judge: To verify whether $M = Unsigncryption(SK_R, PK_S, \sigma)$, the judge runs: 1. steps 1-4 of Unsigncryption in Table 1, 2. if $e(d_2, g_2) \neq e(g, d_3)$, return $\perp$, 3. else if $e(d_1, g) \neq e(g_1, g_R) \cdot e(u_0 \prod_{i=1}^{n} u_i^{c_i}, d_2)$, return $\perp$, 4. else if $M \neq \frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) \cdot e(d_3, g_S)}$, return $\perp$, 5. otherwise return $\top$.

Lemma 1. *Let S_1, S_2 and F be events defined on some probability spaces. Suppose that the event $S_1 \wedge \neg F$ occurs if and only if $S_2 \wedge \neg F$ occurs. Then*

$$| Pr[S_1] - Pr[S_2] | \leq Pr[F].$$

We are now ready to describe the six games.

– **Game 0:** This game is the usual game used to define IND-SCNINR-CCA security. Therefore, the advantage for adversary $\mathcal{A}$ in winning the IND-SCNINR-CCA game is

$$\epsilon = |Pr[\gamma = \gamma' \text{ in game } 0] - 1/2|. \quad (1)$$

– **Game 1:** Game 1 is the same as Game 0, except that the simulator keeps a list of data $(\sigma_0, \sigma_1, \sigma_3, \theta, z, C, g_S, g_R)$ for all unsigncryption and non-repudiation oracles, and he also keeps the data of $(\sigma_0^*, \sigma_1^*, \sigma_3^*, \theta^*, z^*, C^*, y_{S^*}, y_{R^*})$ produced in the challenge oracle.

At the end of the step "oracles after challenge", the simulator checks the whole list to find out whether the following three cases happen:

- Case (1) $(\sigma_0, \sigma_1, g_S) \neq (\sigma_0^*, \sigma_1^*, g_{S^*})$, $\theta = \theta^*$;
- Case (2) $\theta \neq \theta^*$, $z = z^*$;
- Case (3) $z \neq z^*$, $C = C^*$.

If any one of the three cases happens, it aborts.

Analysis: For Case (1) and Case (3), we can find a collision in H_1 and H_2 respectively. For Case (2), we can compute $\log g_3 \leftarrow \frac{\theta - \theta^*}{\sigma_3^* - \sigma_3}$. According to the

previous security definition of H_1, H_2 and D-Log assumption, the possibility for Case (1) to be true is ϵ_{H_1}, Case (2) is ϵ_{Dlog}, and Case (3) is ϵ_{H_2}. Then,

$$Pr[new\ abort\ in\ game\ 1] = \epsilon_{H_1} + \epsilon_{H_2} + \epsilon_{Dlog}. \quad (2)$$

Without this new abort, simulators in Game 0 and Game 1 run in the same manner. Therefore, according to Lemma 1, we have

$$|Pr[\gamma = \gamma'\ in\ game\ 1] - Pr[\gamma = \gamma'\ in\ game\ 0]| \leq Pr[new\ abort\ in\ game\ 1]. \quad (3)$$

Now in Game 1, if the simulator does not abort, then for all unsigncryption and non-repudiation oracles, $C \neq C^*$. This conclusion will be useful for analysis in the latter games. We analyze it from the following four cases:

1. If $(\sigma_0, \sigma_1, y_S) \neq (\sigma_0^*, \sigma_1^*, y_{S^*})$, and since all the above three cases for abort do not happen, then we get $C \neq C^*$.
2. Else if $(\sigma_0, \sigma_1, y_S) = (\sigma_0^*, \sigma_1^*.y_{S^*})$ and $\sigma_3 \neq \sigma_3^*$, then $z \neq z^*$. Since case (3) does not happen, we get $C \neq C^*$.
3. Else if $(\sigma_0, \sigma_1, \sigma_3) = (\sigma_0^*, \sigma_1^*, \sigma_3^*)$, and $g_S = g_{S^*}$, according to the verification equation $e(\sigma_2, g) = e(g_2, g_S) \cdot e(\sigma_1, u_0 \prod_{i=1}^{n} u_i^{c_i})$, we get that $\sigma_2 = \sigma_2^*$ when verification passed. Therefore, in this case $\sigma = \sigma^*$, which is not allowed according to the attack game.
4. Else if $(\sigma_0, \sigma_1, \sigma_3) = (\sigma_0^*, \sigma_1^*, \sigma_3^*)$, and $g_S \neq g_{S^*}$, then $\theta \neq \theta^*$. Case (2) and case (3) do not happen to cause an abort, therefore, $C^* \neq C$.

– **Game 2:** Game 2 is mostly the same as Game 1, with the following three changes:

1. In setup system step, generate $\{g_2, g_3, u_0, U\}$ as follows:
   - Choose random elements $x, y \in \mathbb{Z}_p$, and compute $g_2 \leftarrow g^x$, $g_3 \leftarrow g^y$.
   - To generate U, choose random elements $k_1, ..., k_n \in \mathbb{Z}_p$, and from $i = 1$ to n compute $u_i \leftarrow {g_1}^{k_i}$.
   - To generate u_0, choose random elements $\alpha, \lambda \in \mathbb{Z}_p$, compute $z^* \leftarrow g^\alpha$, $C^* \leftarrow H_2(z^*)$, write C^* as $(c_1^*, ..., c_n^*) \in \{0,1\}^n$. Compute $\tau^* \leftarrow \sum_{i=1}^{n} k_i c_i^*$, then $u_0 \leftarrow {g_1}^{-\tau^*} g^\lambda$.
2. In the challenge step, the simulator generates σ_0^*, σ_1^* according to the signcryption algorithm, but computes σ_2^*, σ_3^* as follows:

$$\sigma_2^* \leftarrow g_{S^*}^x \cdot {\sigma_1^*}^\lambda, \quad \sigma_3^* \leftarrow \frac{\alpha - H_1(\sigma_0^*, \sigma_1^*, g_{S^*})}{y}.$$

3. For all unsigncryption and non-repudiation oracles, if $\sum_{i=1}^{n} k_i c_i = \tau^*$, then the simulator aborts.

Analysis: We now analyze the above three changes one by one.

1. For changes in 1, it is easy to see that $U \in \mathbb{G}^n$, $u_0 \in \mathbb{G}$, $g_2 \in \mathbb{G}$ and $g_3 \in \mathbb{G}$ are still random vector and elements. Therefore, these changes are only notational changes.

2. For changes in 2, if we take $s \leftarrow \frac{\alpha - H_1(\sigma_0^*, \sigma_1^*, g_{S^*})}{y}$, which is also a random element in $\mathbb{Z}_p$, then it is easy to verify that $\sigma_2^* = g_2^{\alpha_{S^*}}(u_0 \prod_{i=1}^n u_i^{c_i^*})^t, \sigma_3^* = s$, which is a valid setting.
3. For changes in 3, recall the conclusion in Game 1 that, if not abort, for all unsigncryption and non-repudiation $C \neq C^*$. And $(k_1, ..., k_n) \in \mathbb{Z}_p^n$ are independent elements chosen randomly by the simulator (independent of the adversary), and for the complexity of discrete logarithm assumption, the value of $(k_1, ..., k_n)$ are computationally hidden from the value of $(u_1, .., u_n)$. Therefore, the value of $(k_1, ..., k_n)$ and independent of the adversary's view. For each unsigncryption oracle and non-repudiation oracle, we have $Pr[\sum_{i=1}^n k_i c_i = \tau^*] = 1/p$.

Finally, we have

$$Pr[\textit{new abort in game } 2] = (q_u + q_n)/p. \tag{4}$$

Without this new abort, the simulator provides the same environment as in Game 1. According to Lemma 1, we have

$$|\, Pr[\gamma = \gamma' \textit{ in game } 2] - Pr[\gamma = \gamma' \textit{ in game } 1]\,| \leq Pr[\textit{new abort in game } 2]. \tag{5}$$

Now in Game 2, if not abort, then for all unsigncryption and non-repudiation oracles, $\sum_{i=1}^n k_i c_i \neq \tau^*$.

– **Game 3:** Game 3 is similar to Game 2, except that in both oracles before challenge step and oracles after challenge step, the simulator computes answers for oracles as follows:

- For each signcryption oracle: Compute $g_2^{\alpha_B} \leftarrow g_B^x$, and signcrypt the message according to the Signcryption algorithm.
- For each non-repudiation oracle: First, run steps 1-4 in unsigncryption algorithm. If $\sum_{i=1}^n k_i c_i = \tau^*$, then the simulator aborts, otherwise it computes $d \leftarrow (d_1, d_2, d_3)$ as follows:

$$(d_1 \leftarrow g_B^{\frac{-\lambda}{\sum_{i=1}^n k_i c_i - \tau^*}}, d_2 \leftarrow g_B^{\frac{-1}{\sum_{i=1}^n k_i c_i - \tau^*}}, d_3 \leftarrow d_2^x).$$

- For each unsigncryption oracle: The simulator first runs the non-repudiation oracle to get d, and then decrypt the signcryptext as follows:

$$M \leftarrow \frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) e(d_3, g_S)}.$$

Analysis: It is easy to verify that

$$d_1 = g_1^{\alpha_B}(u_0 \prod u_i^{c_i})^r, d_2 = g^r, d_3 = g_2^r, \textit{ where } r \leftarrow \frac{-\alpha_B}{\sum_{i=1}^n k_i c_i - \tau^*}.$$

Recall that in Game 2, if $\sum_{i=1}^{n} k_i c_i = \tau^*$, then the simulator also aborts. Therefore, all the changes in the game are just notational. We have:

$$Pr[\gamma = \gamma' \; in \; game \; 3] = Pr[\gamma = \gamma' \; in \; game \; 2]. \tag{6}$$

Now in Game 3, if not abort, the simulator runs the attack game perfectly without the knowledge of α_B.

– **Game 4:** Game 4 is mostly the same as Game 3, except that the simulator tries to embed $BDH = \{g, g^a, g^b, g^c, T \leftarrow e(g,g)^{abc}\}$ (a, b, c are random elements in $\mathbb{Z}_p$) into the simulation by taking the following different steps:
  1. In Setup system step, the simulator sets $g_1 \leftarrow g^a$, $g_B \leftarrow g^b$.
  2. In Challenge step, the simulator computes σ_0^*, σ_1^* as follows:

$$\sigma_0^* \leftarrow e(g,g)^{abc} \cdot M_\gamma, \sigma_1^* \leftarrow g^c.$$

Analysis: If we take $t \leftarrow c$, then we have $\sigma_0^* = e(g_1, g_B)^t \cdot M, \sigma_1^* = g^t$. Since a, b, c are random elements in $\mathbb{Z}_p$, then $g_1 \in \mathbb{G}$, $g_B \in \mathbb{G}$ and $t \in \mathbb{Z}_p$ are also random elements. Therefore, the changes in Game 4 are only notational. Then, we have:

$$Pr[\gamma = \gamma' \; in \; game \; 4] = Pr[\gamma = \gamma' \; in \; game \; 3] \tag{7}$$

Now in Game 4, if not abort, the simulator runs the attack game perfectly with the values of $\{g^a, g^b, g^c, e(g,g)^{abc}\}$, but without the knowledge of (a, b, c).

– **Game 5:** Game 5 represents a slightly modified version of Game 4. Specifically, in this game instead of BDH, the simulator embeds $Random = \{g, g^a, g^b, g^c, T \leftarrow e(g,g)^k\}$ (k is randomly chosen from $\mathbb{Z}_p$) into the simulation by computing $\sigma_0^* \leftarrow e(g,g)^k \cdot M$ in the Challenge step.

Analysis: If the adversary distinguishes the difference between Game 4 and Game 5, then he also distinguishes the two cases of T. From the definition of DBDH assumption, we have:

$$|Pr[\gamma = \gamma' \; in \; game \; 5] - Pr[\gamma = \gamma' \; in \; game \; 4]| \leq \epsilon_{dbdh} \tag{8}$$

For the random and independent choice of T, the adversary's output γ' in this game is independent of the hidden bit γ. We have

$$Pr[\gamma = \gamma' \; in \; game \; 5 | \overline{abort}] = 1/2 \tag{9}$$

Now in Game 5, the simulator aborts with the same probability as in Game 4. If not abort, it simulates Game 5 perfectly with the value of $\{g^a, g^b, b^c, e(g,g)^k\}$, but without the knowledge of (a, b, c, k). According to previous analysis, we can reduce that the simulator aborts in Game 5 with

probability $\epsilon_{H_1} + \epsilon_{H_2} + \epsilon_{Dlog} + (q_u + q_n)/p$, which can be regard as a constant when the times of unsigncryption and non-repudiation oracles are fixed. Therefore, we have

$$Pr[\gamma = \gamma' \text{ in game } 5] = Pr[\gamma = \gamma' \text{ in game } 5|\overline{abort}] \tag{10}$$

Combing all the above formulas in this proof, we get our conclusion that

$$|Pr[\gamma = \gamma' \text{ in game } 0] - 1/2| \leq \epsilon_{H_1} + \epsilon_{H_2} + \epsilon_{Dlog} + (q_u + q_n)/p + \epsilon_{dbdh}. \quad \square$$

Proof of Unforgeability

Theorem 2. *The signcryption scheme is* $(t, q_s, q_u, q_n, \epsilon)$ *SEU-SCNINR-CMA secure, assuming that the Waters signature scheme in [25] is* $(t, q_s, \epsilon/4)$ *existential unforgeable,* H_1 *is* $(t, \epsilon/4)$ *collision resistent,* H_2 *is* $(t, \epsilon/4)$ *collision resistent and the Discrete Logarithm assumption in* $\mathbb{G}$ *holds for* $(t, \epsilon/4)$.

Proof of Theorem 2: In the SEU-SCNINR-CMA game, the adversary $\mathcal{A}$'s goal is to forge a valid signcryptext $\sigma^* = (\sigma_0^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$ where $\sigma^* \neq \sigma^{(i)}$. Throughout this proof, the variables with superscript $^{(i)}$ denote the variables computed in the i-th signcryption oracle. And the variables with superscript * denote the variables computed in the Challenge stage. According to the result of $\mathcal{A}$'s forgery, we divide it into four types as follows:

- Type I: $C^* \neq C^{(i)}$ (for all i form 1 to q_s),
- Type II: $C^* = C^{(i)}$ and $z^* \neq z^{(i)}$ for some $i \in \{1, ..., q_s\}$,
- Type III: $C^* = C^{(i)}$, $z^* = z^{(i)}$ and $\sigma_3^* = \sigma_3^{(i)}$ for some $i \in \{1, ..., q_s\}$,
- Type IV: $C^* = C^{(i)}$, $z^* = z^{(i)}$ and $\sigma_3^* \neq \sigma_3^{(i)}$ for some $i \in \{1, ..., q_s\}$.

We will show that a successful type I forgery will lead to a successful attack on the Waters signature scheme, a successful type II forgery will lead to a break for the collision-resistent hash function H_2, a successful type III forgery will lead to a break of the collision-resistent hash function H_1, and a successful type IV forgery will lead to a solution to the Discrete Logarithm assumption in $\mathbb{G}$.

Before this attack, the simulator $\mathcal{A}'$ flips a random coin to guess which kind of forgery $\mathcal{A}$ will output, then sets up the public parameter and performs appropriately, and all our simulations are perfect.

- **Type I forgery:** We first briefly review the Waters signature scheme [25]. Given a public parameter $Pub_s \leftarrow \{e, \mathbb{G}, \mathbb{G}_T, u_0, U, g, g_2\}$, $\{\alpha_B, g_B \leftarrow g^{\alpha_B}\}$ are computed as private/public key pair of user B (α_B is randomly chosen from $\mathbb{Z}_p$), the signature σ_s on message $C = (c_1, ..., c_n) \in \{0,1\}^n$ is: $(\sigma_{s_0}, \sigma_{s_1}) \leftarrow (g_2^{\alpha_B}(u_0 \prod_{i=1}^n u_i^{c_i})^t, g^t)$. The Waters signature scheme is said to be $(t, q_s, \epsilon/4)$ existential unforgeable (EUF), if given user B's public key g_B, and has access to q_s times signature oracles, the adversary $\mathcal{A}'$ can forge a valid signature on a new message C^* with probability at most $\epsilon/4$.

We let $\mathcal{A}'$ be the simulator of the SEU-SCNINR-CMA game as well as an attacker of existential unforgeability (EUF) game of Waters scheme. $\mathcal{A}'$ will simulate the SEU-SCNINR-CMA game with the knowledge he gets from the EUF game. Next, we show how $\mathcal{A}'$ deals with the simulation as follows:

- In the Setup system step: $\mathcal{A}'$ first gets the public parameter and user B's public key PK_B from the EUF game. Then $\mathcal{A}'$ chooses random $x, y \in \mathbb{Z}_p$, computes $g_1 \leftarrow g^x$, $g_3 \leftarrow g^y$. Finally, $\mathcal{A}'$ runs the SetupPub algorithm to get the other elements of public parameter Pub, and returns Pub and PK_B to $\mathcal{A}$.
- In the Oracles step: $\mathcal{A}'$ is able to answer all the unsigncryption and non-repudiation oracles easily, since $\mathcal{A}'$ can computes $g_1^{\alpha_B} \leftarrow g_B^x$. For signcryption oracles, $\mathcal{A}'$ answers it with the help of signature oracle in EUF game. When $\mathcal{A}$ asks for a signcryption oracle on $(M, PK_S = PK_B, PK_R)$, $\mathcal{A}'$ chooses a random $\alpha \in \mathbb{Z}_p$, computes $C = H_2(g^\alpha)$, and then gets $\sigma_s = (\sigma_{s_0}, \sigma_{s_1})$ on C from the signature oracle. Finally, $\mathcal{A}'$ computes $\sigma_0 = e(\sigma_{s_1}, g_R)^x \cdot M$, $\sigma_1 \leftarrow \sigma_{s_1}$, $\sigma_2 \leftarrow \sigma_{s_0}$, $\sigma_3 \leftarrow (\alpha - H_1(\sigma_0, \sigma_1, g_B))/y$, returns $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$ to $\mathcal{A}$.
- In the Forge step: If $\mathcal{A}$ outputs a successful type I forgery, $\sigma^* = (\sigma_0^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$. Then $\mathcal{A}'$ can also generate a successful forgery $\sigma_s^* \leftarrow \{\sigma_2^*, \sigma_1^*\}$ on a new message $C^* \leftarrow H_2(g^{H_1(\sigma_0^*, \sigma_1^*, g_B)} g_3^{\sigma_3^*})$.

Now we can see that if $\mathcal{A}$ (adversary in SEU-SCNINR-CMA game) finally makes a successful forgery, then $\mathcal{A}'$ (as an attacker of EUF game) also makes a valid forgery for the Waters scheme.

– **Type II forgery:** $\mathcal{A}$ is a type II adversary for the signcryption scheme, $\mathcal{A}'$ is the simulator. Besides, $\mathcal{A}'$ is aimed to find a collision for H_2.

In this case, $\mathcal{A}'$ simulates the game as a normal challenger in the definition. Finally, if $\mathcal{A}$ outputs a successful type II forgery that $C^* = C^{(i)}$ and $z^* \neq z^{(i)}$ for some $i \in \{1, ..., q_s\}$, then $\mathcal{A}'$ finds a collision for hash function H_2.

– **Type III forgery:** $\mathcal{A}$ is a type III adversary for the signcryption scheme, $\mathcal{A}'$ is the simulator. Besides, $\mathcal{A}'$ is aimed to find a collision for H_1.

In this case, $\mathcal{A}'$ simulates the game as a normal challenger in the definition. If $\mathcal{A}$ outputs a successful type III forgery that $C^* = C^{(i)}$, $z^* = z^{(i)}$ and $\sigma_3^* = \sigma_3^{(i)}$ for some $i \in \{1, ..., q_s\}$, then it implies that $\theta^{(i)} = \theta^*$. There are two cases follows:

1. $(\sigma_0^{(i)}, \sigma_1^{(i)}) = (\sigma_0^*, \sigma_1^*)$. According to the check equation $e(\sigma_2, g) = e(g_2, g_S) \cdot e(\sigma_1, u_0 \prod_{i=1}^n u_i^{c_i})$ in the unsigncryption algorithm, we get that if $(\sigma_0^{(i)}, \sigma_1^{(i)}, \sigma_3^{(i)}) = (\sigma_0^*, \sigma_1^*, \sigma_3^*)$, then $\sigma_2^{(i)} = \sigma_2^*$. It is an impossible case, because it contradicts with the requirement of the attack game that $\sigma^{(i)} \neq \sigma^*$.
2. $(\sigma_0^{(i)}, \sigma_1^{(i)}) \neq (\sigma_0^*, \sigma_1^*)$. Then $\mathcal{A}'$ finds a collision in H_1.

– **Type IV forgery:** $\mathcal{A}$ is a type IV adversary for the signcryption scheme, $\mathcal{B}'$ is the simulator. Besides, $\mathcal{A}'$ is given a random element $g_3' \in \mathbb{G}$, and is aimed to compute $y \in \mathbb{Z}_p$ where $g_3' = g^y$.

$\mathcal{A}'$ simulates the game as a normal challenger in the definition except that in the Setup system step, he sets $g_3 \leftarrow g_3'$. Finally, if $\mathcal{A}$ outputs a successful type IV forgery that $C^* = C^{(i)}$, $z^* = z^{(i)}$ and $\sigma_3^* \neq \sigma_3^{(i)}$ for some $i \in \{1, ..., q\}$, then $\mathcal{A}'$ can computes $y \leftarrow (\theta^* - \theta^{(i)})/(\sigma_3^{(i)} - \sigma_3^*)$. □

Proof of Soundness of Non-repudiation

Theorem 3. *The signcryption scheme has perfect soundness of non-repudiation.*

Proof of Theorem 3. In this game, the adversary $\mathcal{A}$ is given the system's public parameter Pub, and he generates a challenge user B's public/privete key pair (PK_B, SK_B). $\mathcal{A}$ is given access to a signcryption oracle. In this oracle, $\mathcal{A}$ outputs a pair of sender/receiver public key (PK_S, PK_B) and a message M, then gets $\sigma \leftarrow Signcryption(SK_S, PK_B, M)$. If the check equation $e(\sigma_2, g) = e(g_2, g_S) \cdot e(\sigma_1, u_0 \prod_{i=1}^{n} u_i^{c_i})$ holds, then the signcryptext σ must be formed as $\sigma = (e(g_1, g_R)^t \cdot M, g^t, g_2^{\alpha_S}(u_0 \prod_{i=1}^{n} u_i^{c_i})^t, s)$ for some $t \in \mathbb{Z}_p$.

Finally, $\mathcal{A}$ outputs a message M' and a non-repudiation evidence d'. If the check equations $e(d_2', g_2) = e(g, d_3')$ and $e(d_1', g) = e(g_1, g_R) \cdot e(u_0 \prod_{i=1}^{n} u_i^{c_i}, d_2')$ both hold, then the non-repudiation evidence d' must be formed as $d' \leftarrow (g_1^{\alpha_R} \cdot (u_0 \prod_{i=1}^{n} u_i^{c_i})^{r'}, g^{r'}, g_2^{r'})$ for some $r' \in \mathbb{Z}_p$. Hence we have

$$M' = \frac{\sigma_0 \cdot e(\sigma_2, d_2')}{e(\sigma_1, d_1')e(d_3', g_S)} = M.$$

It contradicts the hypothesis that $M \neq M'$. Therefore, $\mathcal{A}$ has probability 0 in wining this game. In other words, our proposed scheme satisfies perfect soundness of non-repudiation. □

Proof of Unforgeability of Non-repudiation Evidence

Theorem 4. *The signcryption scheme is* $(t, q_s, q_u, q_n, \epsilon)$ *EUF-NR-evidence - SCNINR-CMA secure, assuming that the Waters signature scheme in [25] is* $(t, q_u + q_n, \epsilon/4)$ *existential unforgeable,* H_1 *is* $(t, \epsilon/4)$ *collision resistent,* H_2 *is* $(t, \epsilon/4)$ *collision resistent and the Discrete Logarithm assumption in* $\mathbb{G}$ *holds for* $(t, \epsilon/4)$.

Proof of Theorem 4. The proof for this theorem is very similar to that for unforgeability. In what follows we highlight key differences between them.

In the EUF-NR-evidence-SCNINR-CMA game, the adversary $\mathcal{A}$'s goal is to forge a valid non-repudiation evidence d^* on σ^* and M^*. According to the result of $\mathcal{A}$'s forgery, we divide it into four types as follows:

- Type I: $C^* \neq C^{(i)}$ (for all i form 1 to $q_u + q_n$),
- Type II: $C^* = C^{(i)}$ and $z^* \neq z^{(i)}$ for some $i \in \{1, ..., q_u + q_n\}$,
- Type III: $C^* = C^{(i)}$, $z^* = z^{(i)}$ and $\sigma_3^* = \sigma_3^{(i)}$ for some $i \in \{1, ..., q_u + q_n\}$,
- Type IV: $C^* = C^{(i)}$, $z^* = z^{(i)}$ and $\sigma_3^* \neq \sigma_3^{(i)}$ for some $i \in \{1, ..., q_u + q_n\}$.

Note that in this proof, the variables with superscript $^{(i)}$ denote the variables computed in the i-th unsigncryption oracle (when $i \leq q_u$) or in the $(i - q_u)$-th non-repudiation oracle (when $q_u < i \leq q_u + q_n$). And the variables with superscript * denote the variables computed in the Challenge stage.

At the beginning of the attack, the simulator $\mathcal{A}'$ firstly flips a random coin to guess which kind of forgery $\mathcal{A}$ will output, then sets up a public parameter and performs appropriately. It turns out that all our simulations are perfect.

Analysis of Type II, III and IV is the same as in the proof of Theorem 2. Therefore, we only analyze Type I forgery and omit analysis for other types here.

- **Type I forgery:** We let $\mathcal{A}'$ be the simulator of the EUF-NR-evidence-SCNINR-CMA game as well as an attacker of existential unforgeablility (EUF) game of Waters scheme. We note that the Waters signature used in this proof has one notational difference from what we have used in the proof of Theorem 2, that is, g_1 is used to replace g_2. Thus, the Waters signature σ_s on message $C = (c_1, ..., c_n) \in \{0,1\}^n$ is: $(\sigma_{s_0}, \sigma_{s_1}) \leftarrow (g_1^{\alpha_B}(u_0 \prod_{i=1}^{n} u_i^{c_i})^t, g^t)$. $\mathcal{A}'$ will simulate the EUF-non-repudiation evidence-SCNINR-CMA game with the knowledge he gets from the EUF game. Next, we show how $\mathcal{A}'$ simulates the game as follows:

  - In the Setup system step: $\mathcal{A}'$ first gets the public parameter and user B's public key PK_B from the EUF game. Then $\mathcal{A}'$ chooses random $x \in \mathbb{Z}_p$, computes $g_2 \leftarrow g^x$. Finally, $\mathcal{A}'$ runs the SetupPub algorithm in signcryption scheme to get the other elements in public parameter Pub, and returns Pub and PK_B to $\mathcal{A}$.
  - In the Orales step: $\mathcal{A}'$ is able to answer the all the signcryption oracles easily, since $\mathcal{A}'$ can computes $g_2^{\alpha_B} \leftarrow g_B^x$. For non-repudiation oracles, A' will answer them with the help of signature oracles in EUF game. When $\mathcal{A}$ asks for a non-repudiation oracle on (σ, PK_S, PK_B), $\mathcal{A}'$ computes C according to the unsigncryption algorithm, and gets $\sigma_s = (\sigma_{s_0}, \sigma_{s_1})$ on C from the signature oracle. Finally, $\mathcal{A}'$ computes $d \leftarrow (\sigma_{s_0}, \sigma_{s_1}, \sigma_{s_1}^x)$. And for each unsigncryption oracle, $\mathcal{A}'$ first runs the non-repudiation oracle to get d, then decrypts $M \leftarrow \frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) e(d_3, g_S)}$.
  - In the Forge step: If $\mathcal{A}$ outputs a successful forgery d^* on (σ^*, M^*) with sender/receiver public keys (PK_{S^*}, PK_B), then $\mathcal{A}'$ can also generate a successful forgery $\sigma_s^* \leftarrow \{d_1^*, d_2^*\}$ for the Waters signature on a new message $C^* \leftarrow H_2(g^{H_1(\sigma_0^*, \sigma_1^*, g_{S^*})} g_3^{\sigma_3^*})$.

  Now we can see that $\mathcal{A}'$ (as an attacker) will finally make a valid forgery for Waters signature scheme, if $\mathcal{A}$ (the adversary in EUF-non-repudiation-evidence-SCNINR-CMA game) makes a successful forgery. □

5 Discussions

5.1 Efficiency Comparison

Our proposed signcryption scheme is based on the signature scheme of Boneh, Shen and Waters[8] (for short, we call it BSW signature). In order to give a better intuition on the comparison of efficiency, we review the BSW signature as follows:

- SetupPub: $Pub_{bsw} = \{\mathbb{G}, \mathbb{G}_T, e, g, g_2, g_3, u_0, U, H_2\}$. Most of the elements in Pub_{bsw} are generated the same way as SetupPub in in Table 1, except that $H_1 : \{0,1\}^* \rightarrow Z_p$.
- KeyGen: The same as KeyGen in Table 1.
- Sign: To sign on $M \in SP_{\mathcal{M}}$, the signer runs almost the same as Signcryption in Table 1, except that there is no σ_0 in the signature and $\theta \leftarrow H_1(\sigma_1, M)$. The signature is $\sigma_{bsw} \leftarrow (\sigma_1, \sigma_2, \sigma_3)$.
- Verify: To verify a signature σ_{bsw} from a signer S, the verifier runs almost the same as Unsigncryption in Table 1, except that it computes $\theta \leftarrow H_1(\sigma_1, M)$ and there is no need to compute M in the last step. If all the check passed, it returns $\top$.

First, we compare our proposed signcryption scheme with the BSW signature scheme on computational cost. From the above description, it is clear the additional cost in signcryption is to compute σ_0 ($\sigma_0 \leftarrow M \cdot e(g_1, g_R)^t$) and the additional cost in unsigncryption is to compute M ($M \leftarrow \sigma_0 / e(\sigma_1, g_1^{\alpha_R})$). Therefore, our signcryptext requires one additional exponentiation in $\mathbb{G}_T$ in signcryption and one additional bilinear computation in unsigncryption, when pre-computation (which will be claimed latter) is applied.

Second, we compare the communication overhead with the BSW signature. In usual communication, the BSW scheme needs to send $(M, \sigma_1, \sigma_2, \sigma_3, ID_S)$, our scheme needs to send $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, ID_S, ID_R)$. When $|M| \approx |\mathbb{G}_T|$, there is nearly no expansion in terms of communication overload (we assume the user ID be a very short string compared with other elements in communication).

Third, we claim that our scheme takes advantage of the the compositional method (either sign-then-encrypt or encrypt-then-sign). For consistency of comparison, we fix the underlying signature scheme as BSW scheme. Since the cost for the compositional method is $1 + 1 = 2$ (that means $Total\text{-}Cost = Cost_{signature} + Cost_{encryption}$), we only has to compare our additional cost with the encryption scheme. For example, we choose the encryption scheme in [7]. The cost for computation cost (if pre-computation applied) is approximately 4 exponentiation in encryption and one bilinear computation in decryption. And the ciphertext size is $2|\mathbb{G}| + |\mathbb{G}_T|$. Clearly, the cost for the encryption scheme is larger than our additional cost.

5.2 Improve Efficiency of the Proposed Scheme

Increase Online Computation Speed. In our scheme, the online computation efficiency can be improved if pre-computation applied. For example, a

sender S can compute $g_2^{\alpha_S}$ and a receiver can compute $g_1^{\alpha_R}$ immediately after the computation of public/private key pair. Then it can be stored for latter use. And when a sender S communicates with a receiver R the first time, the sender S can store the value of $e(g_1, g_R)$, then he does not need to repeatedly compute it in latter communication. Similarly, when a receiver R received a signcryptext from S the first time, he can also store the value of $e(g_2, g_S)$. The judge can also store the value of $e(g_1, g_R)$ and $e(g_2, g_S)$ after the first time of solving computation.

This method costs a little more space for storage, but greatly improves the online computation efficiency. According to an approximate estimation, the online computation time can reduce 56.5% in Signcryption, 25.5% in Unsigncryption, 26% in NR-Evidence-Gen, and 17.3% in JG-verification[2].

Considering that in practise, the cost for storage is cheaper than online computation, the above per-computation method does work on improving the whole efficiency in most cases, except the following two cases. 1. One user communicates with another user once. 2. One judge just deal with repudiation problems between two specific users once.

Reduce the Signcryptext Size. In our original scheme, the signcryptext size is $\sigma \in \mathbb{G}_T \times \mathbb{G}^2 \times \mathbb{Z}_p$. To get a shorter signcryptext, we can replace the symmetric bilinear map with an asymmetric bilinear map [7]: $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and there is an efficiently computable homomorphism $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$. Consider the case where h is a generator of $\mathbb{G}_2$, and $g \leftarrow \varphi(h)$ is a generator of $\mathbb{G}_1$. Then we can get a shorter signcryptext $\sigma \in \mathbb{G}_T \times \mathbb{G}_1^2 \times \mathbb{Z}_p$. The size of the representation of elements in $\mathbb{G}_1$ is $1/k$ of that of $\mathbb{G}_2$, where k is the embedding degree [13]. This method results in lower computation speed, but it leads to a more compact signcryptext and a boarder range of choices of elliptic curve implementations. More details about bilinear maps used in cryptography can be found in [13].

The changes of bilinear maps result in a lot of changes in the scheme, which are shown in detail in Table 3 and Table 4.

The security of this modified scheme is quite similar to the original scheme, except with some small changes corresponding to the change of bilinear maps (from symmetric ones to asymmetric ones).

5.3 Applications of Signcryption with NINR

Signcryption with NINR is suitable for those applications where we assume there will be repudiation disputes between the sender and the receiver. For example, emails, ATM networks, and cryptographic protocols that aims to transport, exchange or establish keys etc.

We take the above mentioned "key" related cryptographic protocols as an example. In such scenarios, since the "key" is a very sensitive message, we normally

[2] We assume for simplicity that a single computation of exponential computation cost one unit of time, a bilinear computation takes 6 units of time, a multi-exponential computation takes 1.5 units of time, and an n-time multiply computation costs one unit of time.

Table 3. SetupPub& KeyGen& Signcryption& Unsigncryption

SetupPub(1^η) by Trusted Party: 1. generate $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ as described above, 2. choose random $h_1, h_2, w_0 \in \mathbb{G}_2$ and a random vector $W \in \mathbb{G}_2^n$, 3. compute the images of elements and vector in step 2 by φ to get $g_1, g_2, u_0 \in \mathbb{G}_1$ and $U \in \mathbb{G}_1^n$. 4. choose a random element $g_3 \in \mathbb{G}_1$, 5. set collision-resistant hash functions $H_1 : \mathbb{G}_T \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_p$, $H_2 : \mathbb{G}_1 \to \{0,1\}^n$. 6. $Pub = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, g_1, g_2, g_3, h_1, h_2, u_0, w_0, U, W, H_1, H_2\}$
KeyGen(Pub, ID_P) by User P: 1. private key for user P is a random $\alpha_P \in \mathbb{Z}_p$, 2. public key for user P is $h_P \leftarrow h^{\alpha_P}$.
Signcryption(SK_S, PK_R, M) by Sender S: To signcrypt $M \in \mathbb{G}_T$ to be communicated to receiver R, sender S runs: 1. steps 1 and 3 of Signcryption in Table 1. 2. compute $\sigma_0 \leftarrow e(g_1, h_R)^t \cdot M$, 3. steps 4-9 of Signcryption in Table 1.
Unsigncryption(SK_R, PK_S, σ) by Receiver R: To unsigncrypt σ from sender S, receiver R runs: 1. steps 1 and 3 of Unsigncryption Table 1. 2. if $e(\sigma_2, h) \neq e(g_2, h_S)e(\sigma_1, w_0 \Pi_{i=1}^n w_i^{c_i})$, return $\perp$, 3. compute $M \leftarrow \sigma_0 / e(\sigma_1, h_1^{\alpha_R})$.

Table 4. NR-Evidence-Gen & JG-Verification

NR-Evidence-Gen$(\sigma, SK_R, PK_S)$by Receiver R: To compute non-repudiation evidence d, receiver R runs: 1. steps 1-2 of Unsigncryption in Table 3, 2. choose a random $r \in \mathbb{Z}_p$, 3. compute $d_1 \leftarrow h_1^{\alpha_R}(w_0 \prod w_i^{c_i})^r$, 4. steps 4-6 of NR-Evidence-Gen in Table 2.
JG-Verification$(\sigma, M, d, PK_S, PK_R)$ by Judge: To verify whether $M = Unsigncryption(SK_R, PK_S, \sigma)$, the judge runs: 1. steps 1-2 in Unsigncryption in Table 3, 2. if $e(g_2, d_2) \neq e(d_3, h)$, return $\perp$, 3. else if $e(g, d_1) \neq e(g_1, h_R) \cdot e(u_0 \prod u_i^{c_i}, d_2)$, return $\perp$, 4. else if $M \neq \frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) \cdot e(d_3, h_S)}$, return $\perp$, 5. otherwise return $\top$.

have the following basic security requirements. From one aspect, the user who generated the "key" (or part of the "key"), should never deny on it, and from the other aspect, the user who exposes the fact that the sender translates such a "key" by a well-formed evidence should also responsible for his act. Fortunately, if we apply signcryption scheme with NINR to construct the cryptographic protocols, soundness of non-repudiation ensures that the non-repudiation evidence d correctly reveals the relationship of a signcryptext σ and a message M, and at the same time, unforgeability of non-repudiation evidence guarantees that the receiver has to be responsible for exposing this relationship if he offered a well-formed evidence.

6 Conclusion

In this work, we propose a model for signcryption with NINR. Compared with the model of Malone-Lee, our model focuses more on the security of NINR by considering two more security requirements. Soundness of non-repudiation makes sure that the property of NINR really works. And unforgeability of evidence data offers a strong requirement for some particular scenarios. Besides, we also come up with a concrete scheme, which is the first signcryption scheme with NINR that can be proved secure without random oracles.

Our scheme should be considered to be a first step in constructing provably secure signcryption with NINR without random oracles. There is still a lot of work that needs to be done. One interesting future research direction relates to efficiency. Our construction makes use of bilinear maps which may take more computational time than that can be afforded in some light applications where low power computing devices dominate. As efficiency is the most important motivation for signcryption, deigning more efficient signcryption schemes with NINR (e.g. avoid using bilinear computations) will be very valuable.

References

1. An, J., Dodis, Y., Rabin, T.: On the Security of Joint Signature and Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
2. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
3. Bao, F., Deng, R.H.: A signcryption scheme with signature directly verifiable by public key. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 55–59. Springer, Heidelberg (1998)
4. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM Journal on Computing 32(3), 586–615 (2003)
5. Baek, J., Steinfeld, R., Zheng, Y.: Formal Proofs for the Security of Signcryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 80–98. Springer, Heidelberg (2002)

6. Bellare, M., Rogaway, P.: Random oracle are practical: A paradigm for designing efficient protocols. In: ACM-CCS 1993, pp. 62–73. ACM press, Fairfax (1993)
7. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security from Identity-Based Techniques. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM-CCS 2005, pp. 320–329. ACM press, Alexandria (2005)
8. Boneh, D., Shen, E., Waters, B.: Strongly Unforgeable Signatures Based on Computational Difie-Hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)
9. Chow, S.S.M., Yiu, S.M., Hui, L.C.K., Chow, K.P.: Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity. In: Lim, J.I., Lee, D.H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 352–369. Springer, Heidelberg (2004)
10. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
11. Damgard, I., Holfheins, D., Kiltz, E., Thorbek, R.: Public-Key with Non-interactive Opening. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 239–255. Springer, Heidelberg (2008)
12. Galindo, D., Libert, B., Fischlin, M., Fuchsbauer, G., Lehmann, A., Manulis, M., Schroder, D.: Public-Key Encryption with Non-Interactive Opening: New Constructions and Stronger Definitions. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 333–350. Springer, Heidelberg (2010)
13. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for Cryptographers, Cryptology ePrint Archive: Report 2006/165, `http://eprint.iacr.org/2006/165`
14. Malone-Lee, J.: A general Construction for Simutaneous Signing and Encrypting. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 116–135. Springer, Heidelberg (2005)
15. Malone-Lee, J.: Signcryption with Non-interactive Non-repudiation. J. Designs, Codes and Cryptography 37(1), 81–109 (2005)
16. Li, F., Shirase, M., Takagi, T.: Efficient Signcryption Key Encapsulation without Random Oracles. In: Yung, M., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, pp. 47–59. Springer, Heidelberg (2009)
17. Libert, B., Quisquater, J.J.: Efficient signcryption with key privacy from gap Diffie-Hellman groups. In: Bao, F., Deng, R.H., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 187–200. Springer, Heidelberg (2004)
18. Libert, B., Quisquater, J.J.: Improved Signcryption from q-Diffie-Hellman Problems. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 220–234. Springer, Heidelberg (2005)
19. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004)
20. Shin, J.B., Lee, K., Shim, K.: New DSA-verifiable signcryption schemes. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 35–47. Springer, Heidelberg (2003)
21. Tan, C.H.: Security Analysis of Signcryption Scheme from q-Diffie-Hellman Problems. J. IEICE Transactions E89-A(1), 206–208 (2006)
22. Tan, C.H.: Insider-secure Hybrid Signcryption Scheme Without Random Oracles. In: ARES 2007, pp. 1148–1154. IEEE Press, Vienna (2007)
23. Tan, C.H.: Insider-secure Signcryption KEM/Tag-KEM Schemes without Random Oracles. In: ARES 2008, pp. 1275–1281. IEEE Press, Barcelona (2008)
24. Toorani, M., Shirazi, A.A.B.: An Elliptic Curve-Based Signcryption Scheme with Forward Secrecy. J. Applied Sciences 9(6), 1025–1035 (2009)

25. Waters, B.: Efficient identity based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
26. Zheng, Y.: Digital signcryption or how to achieve cost (signature&encryption)≪ cost(signature)+cost (encryption). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)

Appendix: A More Efficient Construction in the Random Oracle Model

A.1 The Construction

If random oracle model is allowed, we can construct a modified scheme which is more efficient from all aspects. The public parameter can be reduced from $\mathcal{O}(\log p)$ to $\mathcal{O}(1)$, the size of signcryptext can be reduced from $\mathbb{G}_T \times \mathbb{G}^2 \times \mathbb{Z}_p$ to $\mathbb{G}_T \times \mathbb{G}^2$, and the computational efficiency can also be improved. In this paper, our main goal is to generate signcryption with NINR without random oracles, but we stress that this modified scheme is also meaningful. Since even in the random oracle model, there are no existing signcryption schemes with NINR that fulfilling all the four security requirements of our model.

The main difference is that in the modified scheme $u_0 u_1^{\theta}$ is used to replace $u_0 \prod_{i=1}^{n} u_i^{c_i}$ in the original scheme. The construction is described in Table A-1 and Table A-2, and all the security theorems and proofs for this scheme will be provided in the next subsection.

A.2 Security Proofs

We are going to provide security theorems and proofs for the modified signcryption scheme with NINR in the random oracle model. The difference between the standard model and the random oracle model is that, in the random oracle model, the attacker has access to additional hash oracles in the oracles stage. In this proof, we assume that in each attack game, the attacker can ask for at most q_h time hash oracles on H_1.

Theorem A. 1. *The modified signcryption scheme is* $(t, q_h, q_s, q_u, q_n, \epsilon)$ *IND-SCNINR-CCA secure, assuming that the* (t, ϵ) *DBDH assumption holds, and hash function* H_1 *is a random oracle.*

Proof of Theorem A. 1: We will prove that if $\mathcal{A}$ has advantage ϵ that wins the attack game, then the simulator $\mathcal{A}'$ can solve the DBDH problem with the same advantage ϵ. Initially $\mathcal{A}'$ is given input a tuple (g^a, g^b, g^c, T), T is either g^{abc} or a random element in $\mathbb{G}$.

– In the Setup system stage, $\mathcal{A}'$ sets $g_1 \leftarrow g^a$, the challenge user B's public key $g_B \leftarrow g^b$. Choose random elements $k_1, k_2, \theta^*, \tau \in Z_p$, and compute $u_0 \leftarrow g_1^{-\theta^* \cdot k_1} g^{k_2}$, $u_1 \leftarrow g_1^{k_1}$, $g_2 \leftarrow g^{\tau}$.

Table A-1. SetupPub& KeyGen& Signcryption& Unsigncryption

SetupPub(1^η) by Trusted Party: $Pub = \{\mathbb{G}, \mathbb{G}_T, e, g, g_1, g_2, g_3, u_0, u_1, H_1\}$ is generated the same way as in SetupPub in Table 1. But here we take H_1 as a random oracle.
KeyGen(Pub, ID_P) by User P: The same as KeyGen in Table 1.
Signcryption(SK_S, PK_R, M) by Sender S: To signcrypt $M \in \mathbb{G}_T$ to be communicated to receiver R, sender S runs: 1. steps 1-4 of Signcryption in Table 1, 2. compute $\sigma_2 \leftarrow g_2^{\alpha_S}(u_0 u_1^{\theta})^t$, 3. the signcryptext is $\sigma \leftarrow (\sigma_0, \sigma_1, \sigma_2)$.
Unsigncryption(SK_R, PK_S, σ) by Receiver R: To unsigncrypt σ from sender S, receiver R runs: 1. compute $\theta \leftarrow H_1(\sigma_0, \sigma_1, g_S)$, 2. if $e(\sigma_2, g) \neq e(g_2, g_S) \cdot e(\sigma_1, u_0 u_1^{\theta})$, return $\perp$, 3. otherwise compute $M \leftarrow \sigma_0 / e(\sigma_1, g_1^{\alpha_R})$.

Table A-2. NR-Evidence-Gen & JG-Verification

NR-Evidence-Gen(σ, SK_R, PK_S) by Receiver R: To compute non-repudiation evidence d, receiver R runs: 1. steps 1-2 of Unsigncryption in Table A-1, 2. choose a random $r \in \mathbb{Z}_p$, 3. compute $d_1 \leftarrow g_1^{\alpha_R} \cdot (u_0 u_1^{\theta})^r$, $d_2 \leftarrow g^r$, $d_3 \leftarrow g_2^r$, 4. return $d \leftarrow (d_1, d_2, d_3)$.
JG-Verification$(\sigma, M, d, PK_S, PK_R)$ by Judge: To verify whether $M = Unsigncryption(SK_R, PK_S, \sigma)$, the judge runs: 1. steps 1-2 of Unsigncryption in Table A-1, 2. if $e(d_2, g_2) \neq e(g, d_3)$, return $\perp$, 3. else if $e(d_1, g) \neq e(g_1, g_R) \cdot e(u_0 u_1^{\theta}, d_2)$, return $\perp$, 4. else if $M \neq \frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) e(d_3, g_S)}$, return $\perp$, 5. otherwise return $\top$.

– In the Oracles before challenge stage,
  1. For each hash oracle on $(\sigma_0, \sigma_1, g_S)$, the simulator keeps a list for the input and output for hash oracles (which is initially empty). If the input has already been asked, check the list to find the output, else it returns a random element θ that $\theta \neq \theta^*$, and add $\{(\sigma_0, \sigma_1, g_S), \theta\}$ to the list.
  2. For each signcryption oracle on M with $(PK_S \leftarrow PK_B, PK_R)$, the simulator first computes $g_2^{\alpha_B} \leftarrow g_B^{\tau}$, then it can compute a signcryptext according to the Signcryption algorithm.
  3. For each non-repudiation oracle on σ with $(PK_S, PK_R = PK_B)$, the simulator first runs step 1 of the NR-Evidence-Gen algorithm. If it does not abort, the simulator chooses a random element $r' \in Z_p$, then computes $d_1 \leftarrow g_1^{k_1(\theta-\theta^*)r'} g_B^{\frac{k_2}{k_1(\theta^*-\theta)}} g^{k_2 r'}$, $d_2 \leftarrow g_B^{\frac{1}{k_1(\theta^*-\theta)}} g^{r'}$, $d_3 \leftarrow d_2^{\tau}$. Taking $r \leftarrow \frac{b}{k_1(\theta^*-\theta)} + r'$, then $d_1 \leftarrow g_1^{\alpha_B}(u_0 u_1^{\theta})^r, d_2 \leftarrow g^r, d_3 \leftarrow g_2^r$.
  4. For each unsigncryption oracle on σ with $(PK_S, PK_R = PK_B)$, the simulator first runs the the non-repudiation oracle to get d, then computes $M \leftarrow \frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) e(d_3, g_S)}$.
– In the challenge stage, $\mathcal{A}$ outputs (M_0, M_1) with $(PK_{S^*}, PK_{R^*} = PK_B)$, the simulator computes $\sigma_0^* \leftarrow T \cdot M_\gamma$ (γ is a random bit), $\sigma_1^* \leftarrow g^c$, $\sigma_2^* \leftarrow g_{S^*}^{\tau}$. Finally, it returns $\sigma^* = (\sigma_0^*, \sigma_1^*, \sigma_2^*)$ and then add $\{(\sigma_0^*, \sigma_1^*, g_S), \theta^*\}$ to the hash list.
– In the oracles after challenge stage, the simulator operates similar as in the oracle before challenge stage.
– In the Guess stage, $\mathcal{A}$ outputs a guess bit γ'. If $\gamma = \gamma'$, the simulator outputs a bit 1, or outputs a bit 0 for the DBDH assumption.

If the input tuple is sampled in experiment 0, where $T = e(g,g)^{abc}$, then $|Pr[\gamma = \gamma' \textit{ in experiment } 0] - 1/2| = \epsilon$. Else if the input tuple is sampled from in experiment 1 where $T = e(g,g)^k$, then $Pr[\gamma = \gamma' \textit{ in experiment } 1] = 1/2$. Thus we have

$|Pr[\mathcal{A}' = 1 \textit{ in experiment } 0] - Pr[\mathcal{A}' = 1 \textit{ in experiment } 1]| = |(1/2 \pm \epsilon) - 1/2| = \epsilon$. Therefore, if the adversary $\mathcal{A}$ has advantage ϵ in wining the attack game, then the simulator $\mathcal{A}'$ also has advantage ϵ in solving the DBDH assumption. □

Theorem A. 2. *The signcryption scheme is $(t, q_h, q_s, q_u, q_n, \epsilon)$ SEU-SCNINR-CMA secure, assuming the CDH assumption in $\mathbb{G}$ holds for $(t, \epsilon/q_h)$, and hash function H_1 is a random oracle.*

Proof of Theorem A. 2: We will prove that if $\mathcal{A}$ has advantage ϵ that wins the attack game, then the simulator $\mathcal{A}'$ can solve the CDH problem with advantage at least ϵ/q_h. For CDH assumption, $\mathcal{A}'$ is given input (g^a, g^b), and aims to compute g^{ab}.

– In the Setup system stage, $\mathcal{A}'$ sets $g_2 \leftarrow g^a$, the challenge user B's public key $g_B \leftarrow g^b$. Choose random elements $k_1, k_2, \theta^*, \tau \in Z_p$, and compute $u_0 \leftarrow g_2^{-\theta^* k_1} g^{k_2}$, $u_1 \leftarrow g_2^{k_1}$, $g_1 \leftarrow g^{\tau}$.

– In the Oracles stage,

1. For each hash oracle on $(\sigma_0, \sigma_1, g_S)$, the simulator keeps a list for the input and output for hash oracles (which is initially empty). If the input has already been asked, check the list to find the output. Else it returns $\theta \leftarrow \theta^*$ with probability $1/q_h$, and returns a random element θ that $\theta \neq \theta^*$ with probability $1 - 1/q_h$, and add $\{(\sigma_0, \sigma_1, g_S), \theta\}$ to the list.
2. For each signcryption oracle on M with $(PK_S = PK_B, PK_R)$, the simulator first chooses random elements $t', \theta \in Z_p$, and computes $\sigma_0 \leftarrow e(g_B^{\frac{\tau}{k_1(\theta^*-\theta)}} g_1^{t'}, g_R) \cdot M$, $\sigma_1 \leftarrow g_B^{\frac{1}{k_1(\theta^*-\theta)}} g^{t'}$, $\sigma_2 \leftarrow g_2^{k_1(\theta-\theta^*)t'} g_B^{\frac{k_2}{k_1(\theta^*-\theta)}} g^{k_2 t'}$. Taking $t \leftarrow \frac{b}{k_1(\theta^*-\theta)} + t'$, then $\sigma_0 \leftarrow e(g_1, g_R)^t \cdot M, \sigma_1 \leftarrow g^t, \sigma_2 \leftarrow g_2^{\alpha_S}(u_0 u_1^{\theta})^t$. Finally, the simulator add $\{(\sigma_0, \sigma_1), \theta\}$ the the hash list.
3. For each non-repudiation oracle on σ with $(PK_S, PK_R = PK_B)$, the simulator first computes the $g_1^{\alpha_B} \leftarrow g_B^{\tau}$, then it can compute an answer according to the NR-Evidence-Gen algorithm.
4. For each unsigncryption oracle on σ with $(PK_S, PK_R = PK_B)$, the simulator first runs the the non-repudiation oracle to get d, then computes $M \leftarrow \frac{\sigma_0 \cdot e(\sigma_2, d_2)}{e(\sigma_1, d_1) e(d_3, g_S)}$.

– In the forge stage, if $\mathcal{A}$ outputs a signcryptext $\sigma^* \leftarrow (\sigma_0^*, \sigma_1^*, \sigma_2^*)$ with $(PK_{S^*} = PK_B, PK_{R^*})$, the simulator checks the hash list with input $(\sigma_0^*, \sigma_1^*, g_B)$. If it is not on the input list, then sets the output as θ^*.

If the signcryptext is a valid one, and the output of hash oracle for $(\sigma_0^*, \sigma_1^*, g_B)$ is θ^*, then the simulator can solve the CDH assumption by computing $g^{ab} \leftarrow \sigma_2^*/\sigma_1^{*k_2}$. Now we can see the probability that $\{(\sigma_0^*, \sigma_1^*, g_B), \theta^*\}$ is on the hash list is at least $1/q_h$. Therefore, if $\mathcal{A}$ has advantage ϵ in winning the attack game, then the simulator can solves the CDH assumption with advantage at least ϵ/q_h. □

Theorem A. 3. *The modified scheme has perfect soundness of non-repudiation.*

Proof of Theorem A. 3: In this game, the adversary $\mathcal{A}$ is given the system's public parameter Pub, and he generates a challenge user B's public/privete key pair (PK_B, SK_B). And $\mathcal{A}$ is given access to a signcryption oracle. In this oracle, $\mathcal{A}$ outputs a pair of sender/receiver public key $(PK_S, PK_R = PK_B)$ and a message M, then gets $\sigma \leftarrow Signcryption(SK_S, PK_B, M)$. If the check equation $e(\sigma_2, g) = e(g_2, g_S) \cdot e(\sigma_1, u_0 u_1^{\theta})$ holds, then the signcryptext σ must be formed as $\sigma = (e(g_1, g_R)^t \cdot M, g^t, g_2^{\alpha_S}(u_0 u_1^{\theta})^t)$ for some $t \in \mathbb{Z}_p$.

Finally, $\mathcal{A}$ outputs a message M' and an non-repudiation evidence d'. If the check equations $e(d_2', g_2) = e(g, d_3')$ and $e(d_1', g) = e(g_1, g_R) \cdot e(u_0 u_1^{\theta}, d_2')$ both hold, then the non-repudiation evidence d' must be formed as follows: $d' \leftarrow (g_1^{\alpha_R} \cdot (u_0 u_1^{\theta})^{r'}, g^{r'}, g_2^{r'})$ for some $r' \in \mathbb{Z}_p$.

Hence we have

$$M' = \frac{\sigma_0 \cdot e(\sigma_2, d_2')}{e(\sigma_1, d_1') e(d_3', g_S)} = M.$$

It contradicts the hypothesis that $M \neq M'$. Therefore, $\mathcal{A}$ has probability 0 in wining this game. In other words, this signcryption scheme satisfies perfect soundness of non-repudiation. □

Theorem A. 4. *The modified scheme is* $(t, q_h, q_s, q_u, q_n, \epsilon)$ *EUF-NR-evidence-SCNINR-CMA secure, assuming that CDH assumption in* $\mathbb{G}$ *holds for* $(t, \epsilon/q_h)$*, and hash function* H_1 *is a random oracle.*

Proof of Theorem A. 4: We will prove that if $\mathcal{A}$ has advantage ϵ that wins the attack game, then the simulator $\mathcal{A}'$ can solve the CDH assumption with advantage at least ϵ/q_h. Initially $\mathcal{A}'$ is given input (g^a, g^b).

- In the Setup system stage, $\mathcal{A}'$ sets public parameter as the simulator in the proof of Theorem A.1.
- In the oracles stage, $\mathcal{A}'$ operates similarly as the the simulator in stage of oracles before challenge in the proof of Theorem A.1, except that $\mathcal{A}'$ answers the hash oracles in a different way. For each hash oracle on $(\sigma_0, \sigma_1, g_S)$, it returns $\theta \leftarrow \theta^*$ with probability $1/q_h$, and returns a random element θ that $\theta \neq \theta^*$ with probability $1 - 1/q_h$,
- In the forge stage, $\mathcal{A}$ outputs $(d^*, \sigma^*, M^*, PK_{S^*}, PK_{R^*})$ with $PK_{R^*} = PK_B$. The simulator checks the hash list, if $(\sigma_0^*, \sigma_1^*, g_{S^*})$ is not on the hash list as input, then adds $\{(\sigma_0^*, \sigma_1^*, g_S), \theta^*\}$ to the list.

If d^* is a valid one, and $\{(\sigma_0^*, \sigma_1^*, g_{S^*}), \theta^*\}$ is on the hash list, then the simulator can solve the CDH assumption by computing $g^{ab} \leftarrow \sigma_2^*/\sigma_1^{*k_2}$. Now we can see the probability that $\{(\sigma_0^*, \sigma_1^*), \theta^*\}$ is on the hash list is at least $1/q_h$. Therefore, if $\mathcal{A}$ has advantage ϵ in winning the attack game, then the simulator can solves the CDH assumption with advantage at least ϵ/q_h. □