Non-interactive Deniable Authentication Protocols

Haibo Tian[1], Xiaofeng Chen[2], and Zhengtao Jiang[3]

[1] School of Information Science and Technology, Sun Yat-Sen University, Guangzhou, 510275, China
`sysutianhb@gmail.com`

[2] School of Telecommunications Engineering, Xidian University, Xi'an, 710071, China
`xfchen@xidian.edu.cn`

[3] School Of Computer, Communication University of China, Beijing, 100024, China
`z.t.jiang@163.com`

Abstract. This paper gives a security model for non-interactive deniable authentication (NIDA) protocols. This model captures a session-state-reveal attack and a key-compromise-impersonation attack. We analyze some NIDA protocols in the model, and we find that no one is satisfactory. We give a new paradigm to construct an NIDA protocol, which is provably secure in the model.

Keywords: Deniable Authentication, Non-interactive Protocols, security model.

1 Introduction

The deniable authentication (DA) [1] means that a sender not only proves that she/he is the communicating entity with a receiver but also leaves no evidence to the receiver about the participating in a protocol. The deniability feature is desirable in some applications, such as the Off-the-Record Messaging [2], and the Internet Key Exchange protocol [3].

To make a DA protocol more efficient, Shao [4] proposed a concept of non-interactive deniable authentication (NIDA). It claims that a one-pass transcript is enough for the goal of a DA protocol. However, a simple replay attack can be used to falsify the authentication goal of most NIDA protocols. A common countermeasure is to require that each message includes a time stamp by default.

Another problem about NIDA is a gap between its security model and analysis techniques.

- The security model of the NIDA is similar to that of a designated verifier signature (DVS). There is no security model at the beginning [4]. Wang and Song [5] proposed a formal model which is similar to the model of DVS.
- The analysis techniques consider various attacks. The identified attacks include an impersonation attack [4], a session-state-reveal (SSR) attack [6],

a key-compromise-impersonation (KCI) attack [7], and a man-in-the-middle (MITM) attack [8]. An impersonation attack means that an adversary can impersonate an intended receiver to identify the source of a given message. An SSR attack means that an adversary can forge a message if some session-specific values are compromised. A KCI attack means that an adversary who knows a sender's private key can impersonate an honest receiver to identify the source of a message produced by the sender. An MITM attack means that an adversary can establish a session key with either a sender or a receiver.
- The gap is that the security model cannot capture these attacks. The SSR attack and KCI attack are not considered in the model [5]. Sometimes, these attacks are considered by independent proofs [6,7]. The problem is that there is no clear description about an adversary.

This paper focuses on the security model of NIDA protocols. We present a model and analyze some NIDA protocols. Then we give a paradigm to construct satisfactory NIDA protocols.

1.1 Related Works

Lu et al. [9] and Willy et al. [10] proposed protocols similar to a ring signature [11]. The protocol of Lu et al. took a receiver into a signer ring to achieve deniability. The protocol of Willy et al. used chameleon hash functions to separate messages and signatures. Wang et al. [5] proposed a scheme based on the DVS where a simulation procedure was used to achieve deniability. There are many protocols based on a message authentication code (MAC) [6, 8, 12–22]. Generally, there is an MAC key to protect a message. Since the key can be calculated by a receiver or by both a sender and a receiver, the deniability property is achieved.

1.2 Contributions

- **Security Model:** A new model is based on that of DA protocols in [23]. An adversary in the model can deliver messages, corrupt entities and reveal session-state values. There are definitions about authenticator, transcript deniability, and full deniability. There is also a message identification (MI) protocol that is an idea NIDA protocol.
- **Protocol Analysis:** We run some NIDA protocols in the new model. We give a table to summarize their satisfactions to each definitions. There is no satisfactory NIDA protocols. We detail two attacks to show an analysis method in the model.
- **New Paradigm:** A new paradigm emulates an MI protocol in the model. We give a concrete scheme based on the Rabin signature [24]. It is provably secure in the model.

1.3 Organizations

Section 2 is some preliminaries, including some assumptions and a traditional description of NIDA protocols. Section 3 is the new security model, and some analysis about NIDA protocols. Section 4 is the new paradigm to construct NIDA protocols. The concrete scheme is shown in section 5. The comparison is in section 6. The last section concludes the paper.

2 Preliminaries

2.1 Assumptions

– *Computational Diffie-Hellman (CDH) problem:* Given large primes p, q satisfying $q|p-1$, there is a generator $g \in \mathbb{Z}_p^*$ for a group $\mathbb{G}$ with an order q. Given two random elements $g^x, g^y \in_R \mathbb{G}$, the problem is to find $g^z \in \mathbb{G}$ such that $z = xy \bmod q$.
– *Decisional Diffie-Hellman (DDH) problem:* With the same parameters $(p, q, g, \mathbb{G})$, given three random elements g^x, g^y, g^z, the problem is to decide whether $z = xy \bmod q$.

The assumption is that there are no polynomial time algorithms to solve a CDH (DDH) problem with non-negligible probability ϵ in time t when q is big enough.

2.2 NIDA Protocols

An NIDA protocol includes four algorithms $(Setup, Prove, Verify, Sim)$.

– On input of a security parameter $k \in \mathbb{N}$, a $Setup$ algorithm generates system parameters and public/private key pairs. The key pair of a sender is usually denoted by (pk_S, sk_S), and a receiver by (pk_R, sk_R);
– A $Prove$ algorithm takes as input a message m, the public key pk_R, the secret key sk_S to generate an authenticator $authen$. Then the sender sends $c = m||authen$ to the receiver;
– A $Verify$ algorithm takes as input a transcript c, the public key pk_S and the secret key sk_R to produce a decision bit $b \in \{0, 1\}$, where $b = 1$ means that the receiver accepts;
– A Sim algorithm takes as input the public key pk_S, the secret key sk_R, and a message m to generate a simulated transcript $\hat{c}$ which is computationally indistinguishable from a real transcript c associated with the given message m.

There are some properties about NIDA protocols. We adopt the descriptions in [5, 8, 25] with some modifications.

1. *Correctness:* If a sender and a receiver follow the description of an NIDA protocol, the receiver is always able to identify the source of the message in a transcript, which means the receiver accepts.

2. *Unforgeability:* An adversary cannot generate a new valid transcript in polynomial time when the adversary can obtain public keys and some qualified transcripts, where messages are determined by the adversary.
3. *Deniability:* An adversary cannot distinguish a simulated transcript from a real one in polynomial time even the adversary can obtain public keys, some real transcripts and simulated transcripts for the adversary's messages.
4. *Resistance to impersonation attack:* An adversary cannot impersonate a qualified receiver to identify the source of a message in a transcript even the adversary can get access to public keys and valid transcripts.
5. *Resistance to SSR:* A disclosed session-specific value does not affect the secure properties of other sessions of an NIDA protocol. Note that a session means one interaction between a sender and a receiver.
6. *Resistance to man-in-the-middle attack:* An adversary cannot establish session keys with either a sender or a receiver even the adversary controls all communication channels between the sender and receiver.
7. *Resistance to KCI attack:* An adversary cannot impersonate a qualified receiver, even if the adversary can get access to the private key of a sender.

3 The Security Model

The current security model [5] does not capture some common attacks to the NIDA protocols. We here give a new model. It is based on an extension framework of Raimondo et al. [23].

Message Driven Non-interactive Protocols. A non-interactive protocol is a process that is initially invoked by a party with some initial state. Once invoked, the protocol waits for an activation that can happen for a message from the network or an external request. Upon activation, the protocol processes the incoming data together with its current internal state generating an outgoing transcript and/or an output. Once the activation is completed, the protocol finishes.

The Authenticated-Links Model (AM). There are n parties $P_1, \ldots, P_n$, each running a copy of a message-driven protocol π. The computation consists of an activation of π within different parties. The adversary $\mathcal{A}$ is a probabilistic polynomial-time (PPT) algorithm with the following abilities:

- *control and schedule activations:* $\mathcal{A}$ can decide who is the next party to activate and which incoming message or external request the activated party is to receive.
- *deliver messages:* $\mathcal{A}$ can change the order of delivery and can choose not to deliver some messages at all. However, $\mathcal{A}$ is restricted to deliver messages faithfully. That is, we assume that each message carries the identities of the sender P_i and of the intended receiver P_j. When a message is sent by a party, it is added to a set M of authentic messages. Whenever $\mathcal{A}$ activates a party P_j on some incoming message m, it must be that m is in the set M and that P_j is the intended receiver of m.

- *corrupt parties:* $\mathcal{A}$ learns the entire current state of the corrupted party P_i and can add to the set M any fake messages on behalf of P_i. A special symbol in the output of P_i is generated to signal the corruption. $\mathcal{A}$ will control all the sequent activations of P_i.

In addition, on the completion of an activation, the outgoing messages, external requests and the output generated by the protocol become known to $\mathcal{A}$. We refer to such an adversary as an AM-adversary.

With all honest parties and the AM-adversary, there is a global output of a running protocol. Let $AUTH_{\pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{r})$ denote the global output of a running of the protocol π with the n parties and the adversary $\mathcal{A}$ with input $\boldsymbol{x} = x_1, \ldots, x_n$ and random input $\boldsymbol{r} = r_0, r_1, \ldots, r_n$, where r_0 is for $\mathcal{A}$ and x_i and r_i are for a party P_i, $i > 0$. Let $AUTH_{\pi,\mathcal{A}}(\boldsymbol{x})$ denote the random variable describing $AUTH_{\pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{r})$ when $\boldsymbol{r}$ is uniformly chosen.

Remark 1. Note that the message set M is named authentic messages. The messages are not deleted after a reception. This is due to the nature of non-interactive protocols.

Remark 2. The corrupt ability captures the KCI attack.

The Unauthenticated-Links Model (UM). The computation of unauthenticated-links model is similar to the AM model but the restriction of delivering messages faithfully is removed for the adversary $\mathcal{U}$, referred to as an UM-adversary. Instead, it can deliver arbitrary messages. Besides this, we give the adversary $\mathcal{U}$ an ability to obtain secret information from an honest party's internal state.

- *session-state reveal (SSR):* $\mathcal{U}$ can learn some values in the current state of an uncorrupted party before or after an activation is completed. The *restriction* is that the disclosure of the session-specific values cannot lead to the exposure of the party's long term private key.

Further,there is an initialization function I that models an initial phase out-of-band and authenticated information exchange between the parties.

The random variables $UNAUTH_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r})$ and $UNAUTH_{\pi,\mathcal{U}}(\boldsymbol{x})$ are defined analogously to the previous ones $AUTH_{\pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{r})$ and $AUTH_{\pi,\mathcal{A}}(\boldsymbol{x})$, but with the computation carried out in the unauthenticated-links model.

Remark 3. Note that the restriction of the SSR ability makes the ability different to the corrupt ability. The SSR ability captures the SSR attack.

Emulation of Protocols. When we say that a protocol π' in the unauthenticated-links model emulates a protocol π in the authenticated-links model, we want to capture the idea that the running π' in an unauthenticated network has the same effect as the running π in an authenticated network. Formally speaking:

Definition 1. *Let π' and π be the message-driven protocols for n parties. We say that π' emulates π in unauthenticated networks if for any UM-adversary $\mathcal{U}$ there exists an AM-adversary $\mathcal{A}$ such that for all inputs $\boldsymbol{x}$,*

$$AUTH_{\pi,\mathcal{A}}(\boldsymbol{x}) \overset{c}{=} UNAUTH_{\pi',\mathcal{U}}(\boldsymbol{x}) \tag{1}$$

where $\overset{c}{=}$ denotes computationally indistinguishable.

Authenticators. An authenticator is a *compiler* that takes as input protocols designed for authenticated networks, and turns them into *equivalent* protocols for unauthenticated networks.

Definition 2. *A compiler $\mathcal{C}$ is an algorithm that takes as input descriptions of protocols and produces descriptions of protocols. An authenticator is a compiler $\mathcal{C}$ where for any protocol π, the protocol $\mathcal{C}(\pi)$ emulates π in unauthenticated networks.*

In particular, authenticators translate secure protocols in the authenticated-links model into secure protocols in the unauthenticated-links model. The simplest protocol is a message identification (MI) protocol that transports a message from a party to another for identification. It can be described formally as follows:

- On activation within P_i on external request (P_j, m), the party P_i sends the message (P_i, P_j, m) to party P_j and outputs 'P_i sent m to P_j';
- Upon receipt of a message (P_i, P_j, m), P_j outputs 'P_j identified the source of m as P_i'.

A protocol that emulates the above MI protocol in unauthenticated-links model is called an *MI-authenticator*.

Remark 4. The MI-authenticator captures the unforgeability property.

Definition 3. *An MI-authenticator λ is transcript deniable if there exists a simulator S_λ that given a message m sent by a party A to B produces a transcript of a session of λ for m, which is computational indistinguishable from a real one for an adversary who does not corrupt the party B or send this real or simulated transcript to B, and not reveal the session-state of S_λ for this transcript or that of A for the real one.*

Definition 4. *An MI-authenticator λ is full deniable if there exists a simulator $S_\lambda^{pri_B}$ accessing the private key of B that given a message m sent by a party A to B produces a transcript of a session of λ for m, which is computational indistinguishable from a real one for an adversary who does not reveal the session-state of $S_\lambda^{pri_B}$ for this transcript or that of A for the real one.*

Remark 5. The two definitions capture the deniability property of NIDA protocols. The transcript deniability means that an adversary can not distinguish a sender from infinite possible senders. The full deniability means that an adversary can not distinguish a sender from a receiver.

3.1 Protocol Analysis

We give two examples to show an analysis method in the model.

The Protocol of Lee et al. [6] There is a report about the protocol [25]. It reported that the protocol [6] was not KCI secure since an adversary with the private key of a receiver could impersonate a sender. However, this is just the full deniability. It is meaningless to consider this attack if we take the full deniability as a desirable property.

We here give a real attack to show that the scheme is not secure under the SSR attack.

- **The Protocol**
  - **Setup** The system parameter is $(p, q, g, \mathbb{G}, H)$, where $H : \{0,1\}^* \to \mathbb{Z}_q$ and other symbols are the same as those in Section 2.1. For a sender S, $sk_S \in_R \mathbb{Z}_q$ and $pk_S = g^{sk_S} \bmod p$. For a receiver R, (sk_R, pk_R) is computed similarly.
  - **Prove** Select $r \in_R \mathbb{Z}_q$, compute $\Lambda = g^r \bmod p$ and
  $$MAC = H((pk_R)^{H(m)sk_S + r\Lambda \bmod q} \bmod p||m),$$
  where "$||$" denotes bits concatenation. The transcript is $c = (m, \Lambda, MAC)$
  - **Verify** Verify whether $H((pk_S^{H(m)}\Lambda^\Lambda)^{sk_R} \bmod p||m) = MAC$.
- **The Attack**
  - An adversary $\mathcal{A}$ sends a transcript (m, Λ, MAC) to an honest receiver. It reveals a session key $sk = (pk_S^{H(m)}\Lambda^\Lambda)^{sk_R}$ of the receiver.
  - Then $\mathcal{A}$ modifies m arbitrarily to obtain $m' \neq m$ and sends (m', Λ, MAC) to the same receiver. It reveals another session key $sk' = (pk_S^{H(m')}\Lambda^\Lambda)^{sk_R}$.
  - $\mathcal{A}$ computes $g^{sk_S sk_R} = (k/k')^{(H(m)-H(m'))^{-1}}$.
  - $\mathcal{A}$ produces a transcript $(m_\mathcal{A}, \Lambda_\mathcal{A}, MAC_\mathcal{A})$ where $m_\mathcal{A}$ is an arbitrary message, $\Lambda_\mathcal{A} = pk_S^\alpha$ for a random value $\alpha \in_R \mathbb{Z}_q$, and $MAC_\mathcal{A} = H(k_\mathcal{A}||m_\mathcal{A})$ where $k_\mathcal{A} = (g^{sk_S sk_R})^{(H(m_\mathcal{A})+\alpha\Lambda_\mathcal{A} \bmod q)} \bmod p$.
  - The forged transcript can be accepted according to the protocol description. Thus, the protocol is not an MI-authenticator.

The Protocol of Fan et al. [18] Besides the session key, other session-specific values may also help an adversary. The protocol in [18] is suitable to show the help.

- **The protocol.** We omit the message encryption part of the scheme as it is not related to the unforgeability property.
  - **Setup** A key generation center (KGC) sets groups $(G_1, +)$ and $(G_2, \cdot)$ with order p. The generator of G_1 is P. A paring is $e : G_1 \times G_1 \to G_2$. The KGC randomly selects $s \in_R \mathbb{Z}_p^*$ and sets $P_{pub} = sP$. Three hash functions

are $H_1 : \{0,1\}^* \to G_1$, $H_2 : G_2 \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ and $H_3 : G_2 \times \{0,1\}^* \to \mathbb{Z}_p^*$. The KGC computes a user's private key as $D_{ID} = sH_1(ID)$ where ID is the user's identity.

- **Prove** A sender, ID_S, computes $Q_R = H_1(ID_R)$ and $\delta = e(Q_R, Q_R)^r$ for $r \in_R \mathbb{Z}_p^*$. Then the sender computes $\Lambda = H_2(\delta, T)$ where $T \in \mathbb{Z}_p^*$ is a timestamp, and $U = rQ_R - \Lambda D_{ID_S}$, and $MAC = H_3(\delta, m)$. The protocol transcript is $c = (ID_S, \Lambda, U, T, MAC, m)$.
- **Verify** The receiver checks the validity of the timestamp. Then the receiver computes $\delta' = e(U, Q_R)e(Q_S, D_{ID_R})^{\Lambda}$ and verifies whether $\Lambda = H_2(\delta', T)$ and $MAC = H_3(\delta', m)$.

– **The Attack**

- $\mathcal{A}$ reveals the value δ of a session and computes

$$e(Q_S, Q_R)^s = (\delta / e(U, Q_R))^{\Lambda^{-1}}.$$

- $\mathcal{A}$ requests ID_S to send a transcript $(ID_S, \Lambda_S, U_S, T_S, MAC_S, m_S)$.
- $\mathcal{A}$ computes $\delta_{\mathcal{A}} = e(U_S, Q_R)(e(Q_S, Q_R)^s)^{\Lambda_S}$. A forged transcript is $(ID_S, \Lambda_S, U_S, T_S, MAC', m')$, where m' is an arbitrary message and $MAC' = H_3(\delta_{\mathcal{A}}, m')$.
- The forged transcript can be accepted according to the protocol. So the protocol is not an MI-authenticator.

Other protocols are analyzed similarly. We give a table in the Section 6 as a summary and comparison.

4 A New Paradigm

4.1 Selectively Unforgeable But Existentially Forgeable Signatures

In the new paradigm, we use a general signature scheme $(KGen, Sign, Ver)$. The scheme is existentially forgeable but not selectively forgeable. The existentially forged signature should be indistinguishable from a real signature. We define its security using the following game. We assume a simulator Sim and a PPT forger $\mathcal{F}$. They play a game as follows:

1. Sim runs $KGen$ to produce a key pair (sk_S, pk_S) for a signer and gives pk_S to $\mathcal{F}$.
2. Sim produces a challenge message m^* and gives it to $\mathcal{F}$;
3. $\mathcal{F}$ produces a signature δ^* for m^*;
4. The forger wins if the pair (m^*, δ^*) is qualified.

We define that a signature scheme is selectively unforgeable but existentially forgeable (SUEF) if a forger cannot win the above game in polynomial time t with a non-negligible probability ϵ, and a PPT adversary cannot distinguish an existentially forged signature from a real one.

4.2 The Construction

We construct a deniable MI-authenticator λ_{DMI} as follows:

– The initialization function I invokes a group generation algorithm to produce the parameters (p, q, g) for a group $\mathbb{G}$, where $p = 2q + 1$, and $g \in_R \mathbb{Z}_p^*$ is a generator with an order q. Let E denote a symmetric encryption algorithm, such that for any key k_e of length l, the function E_{k_e} is a permutation over b-bit strings. Let $H : \{0,1\}^* \to \{0,1\}^l$ be secure hash functions.

  Then I invokes, once for each party, the *KGen* algorithm to produce key pairs (sk_i, pk_i) for P_i. Each party is assigned a secrete trapdoor key $t_i \in_R \mathbb{Z}_q$ and a public trapdoor key $T_i = g^{t_i}$.

  The public information is the system parameters, all public keys and all public trapdoor keys: $I_0 = (p, q, g, H, E, pk_1, T_1, \ldots, pk_n, T_n)$. The private information for P_i is $I_i = (sk_i, t_i)$.

– When activated, within party P_i and with an external request (m, P_j), the protocol λ_{DMI} invokes a two party protocol $\hat{\lambda}_{DMI}$ that proceeds as follows. P_i sends a transcript: $m, P_i, e = (T_j)^r, \delta = Sign_{sk_i}(E_{H(m,P_i,P_j)}(g^r))$' to P_j, where $r \in_R \mathbb{Z}_q$. Then P_i produces an output 'P_i sent m to P_j'.

– Upon receipt of 'message: m, P_i, e, δ', party P_j computes $g^r = e^{t_i^{-1}}$ and verifies whether the signature δ is valid for the value $E_{H(m,P_i,P_j)}(g^r)$. If the verification is true, party P_j outputs 'P_j identified the source of m as P_i'.

Pictorially, the protocol is described in Fig.1.

$$
\begin{array}{lcl}
\qquad P_i & & \qquad\qquad P_j \\
r \in_R \mathbb{Z}_q & & \\
e \leftarrow (T_j)^r & & \\
\delta \leftarrow Sign_{sk_i}(E_{H(m,P_i,P_j)}(g^r)) & & \\
 & \xrightarrow{m, P_i, e, \delta} & \\
 & & \left.\begin{array}{l} true \\ false \end{array}\right\} \leftarrow Ver_{pk_i}(\delta, E_{H(m,P_i,P_j)}(e^{t_j^{-1}}))
\end{array}
$$

Fig. 1. A new paradigm for the non-interactive deniable authentication

4.3 The Proofs

Proposition 1. *If the signature scheme is (t_s, ϵ_s) SUEF secure, the CDH problem is (t_c, ϵ_c) hard, and the symmetric encryption E and decryption E^{-1} are modeled by random oracles, the protocol λ_{DMI} emulates the protocol MI in the unauthenticated-links network.*

Proof. Let $\mathcal{U}$ be an UM-adversary that interacts with λ_{DMI}. We construct an AM-adversary $\mathcal{A}$ such that $AUTH_{MI,\mathcal{A}}() \overset{c}{=} UNAUTH_{\lambda_{DMI},\mathcal{U}}()$.

Adversary $\mathcal{A}$ runs $\mathcal{U}$ on a simulated interaction with a set of parties running λ_{DMI}.

- $\mathcal{A}$ chooses and distributes keys for the imitated parties, according to function I.
- When $\mathcal{U}$ activates an imitated party A' for sending a message (m, A', e, δ) to an imitated party B', the adversary $\mathcal{A}$ activates the party A in the authenticated network to send m to B.
- When an imitated party B' produces 'B' identified the source of $\hat{m}$ as A'', the adversary $\mathcal{A}$ activates the party B in the authenticated-links model with the incoming message $\hat{m}$ from A.
- When $\mathcal{U}$ corrupts a party, $\mathcal{A}$ corrupts the same party in the authenticated network and hands the corresponding information from the simulated run to $\mathcal{U}$.
- When $\mathcal{U}$ reveals the session state, $\mathcal{A}$ hands the values from the internal states of imitated parties to $\mathcal{U}$. Finally, $\mathcal{A}$ produces whatever $\mathcal{U}$ produces.

Let $\mathcal{B}$ denotes the event that the imitated party B' produces 'B' identified the source of $\hat{m}$ as A'' where A' and B' are uncorrupted and the message $(\hat{m}, A, B)$ is not currently in the authentic message set M. This implies that A was not activated for sending $\hat{m}$ to B. If the event $\mathcal{B}$ never happens, the simulation of $\mathcal{A}$ is perfect and $AUTH_{MT,\mathcal{A}}() \overset{c}{=} UNAUTH_{\lambda_{DMT},\mathcal{U}}()$.

It remains to show that event $\mathcal{B}$ occurs only with negligible probability. Assume the event $\mathcal{B}$ occurs with probability ϵ within time t. We construct a forger $\mathcal{F}$ that breaks the underlying signature scheme or solves a CDH problem. The forger $\mathcal{F}$ interacts with Sim as specified in Section 4.1 to obtain (pk_S, m^*). The forger $\mathcal{F}$ also gets a CDH problem instance (g^x, g^y). The strategy of $\mathcal{F}$ is to run the adversary $\mathcal{U}$.

$\mathcal{F}$ provides $\mathcal{U}$ two random oracles O_E and O_E^{-1} for the computations of E and E^{-1}.

- O_E maintains an E_{list} recording all inputs and outputs. The input to O_E includes a message m, identities of a sender P and a receiver Q, and an element $R \in \mathbb{G}$. If the input is not in the E_{list}, O_E randomly selects a value $c \in \{0,1\}^{|p|}$ as an output, where $|p|$ denotes the bit length of p. The E_{list} is updated by adding the input-output record (m, P, Q, R, c).
- O_E^{-1} maintains an R_{list} which is empty at the beginning. O_E^{-1} takes as input (m, P, Q, c). If there is a match entry in the E_{list} indexed by the input,

O_E returns the value R. Else O_E^{-1} produces an output $R = g^r \in \mathbb{G}$ where $r \in_R \mathbb{Z}_q^*$. O_E^{-1} adds a record (m, P, Q, R, c) in the E_{list} and a record (R, r) in the R_{list}.

Now we specify the game between the forger $\mathcal{F}$ and the adversary $\mathcal{U}$.

- $\mathcal{F}$ runs the function I to set parameters and keys for a set of imitated parties who run the protocol λ_{DMI}. Then the public verification key associated with some party P^*, chosen at random, is replaced by the key pk_S. $\mathcal{F}$ gives all public information to $\mathcal{U}$.
- If during the simulation, P^* is queried to generate a transcript for a message m to a party Q, the forger $\mathcal{F}$ existentially forges a message-signature pair (m_F, δ_F) w.r.t. the key pk_S. Then $\mathcal{F}$ queries the O_E^{-1} oracle with (m, P^*, Q, m_F) to get a reply R. $\mathcal{F}$ computes $e = R^{t_Q}$ and replies to the adversary $\mathcal{U}$ the transcript (m, P^*, e, δ_F).
- Other message delivery queries are responded according to the protocol specification with the oracle access to O_E.
- If $\mathcal{U}$ corrupts a party, the private key of the party is given to $\mathcal{U}$. If $\mathcal{U}$ corrupts P^*, $\mathcal{F}$ fails.
- If $\mathcal{U}$ queries to reveal the session states of one run of a party, $\mathcal{F}$ gives the value r of that run to $\mathcal{U}$. $\mathcal{F}$ will find a value r in the R_{list} as a response if the party is P^*.

If a party Q^* is uncorrupted, and outputs 'Q^* identified the source of m as P^*' but P^* was not activated to send m to Q^*, the forger $\mathcal{F}$ finds the last message received by Q^* such as $(m, P^*, e^* = (T_{Q^*})^{r^*}, \delta^* = Sign_{sk_{P^*}}(E_{H(m,P^*,Q^*)}(g^{r^*})))$. Then $\mathcal{F}$ tries to find a match in the R_{list} indexed by $e^{*t_{Q^*}^{-1}}$.

- If there is no match in the R_{list}, $\mathcal{F}$ rewinds $\mathcal{U}$ to the point where $(m, P^*, Q^*, e^{*t_{Q^*}^{-1}})$ is queried to the oracle O_E. This time the forger $\mathcal{F}$ sets the output of O_E as $c = m^*$. Then the forger $\mathcal{F}$ runs $\mathcal{U}$ again. According to the general forking lemma [26], there is a non-negligible probability for $\mathcal{U}$ to produce another qualified signature for the query $(m, P^*, Q^*, e^{*t_{Q^*}^{-1}})$. The signature is returned to Sim as a response by the forger $\mathcal{F}$.
- If there is a match in the R_{list}, $\mathcal{F}$ resets the public key of P^* as g^x and runs $\mathcal{U}$ again. If a party Q' is uncorrupted, and outputs 'Q' identified the source of m' as P^*' but P^* was not activated to send m' to Q', the forger $\mathcal{F}$ finds out the last message received by Q' such as $(m', P^*, e' = (T_{Q'})^{r^*}, \delta^* = Sign_{sk_{P^*}}(E_{H(m',P^*,Q')}(g^{r^*})$. Then $\mathcal{F}$ finds the match (R', r') in the R_{list} indexed by $e'^{t_{Q'}^{-1}}$ and the match (m', P^*, Q', R', c') in the E_{list} indexed by (m', P^*, Q', R').

  Then $\mathcal{F}$ rewinds $\mathcal{U}$ to the point when (m', P^*, Q', c') is queried to the oracle O_E^{-1}. This time the forger $\mathcal{F}$ sets the output of O_E^{-1} as g^y. Note that P^* was not activated to send m' to Q'. There is no session state about (m', P^*, Q') in the party P^*. So there is no impact on the session state real quires of $\mathcal{U}$. Then the forger $\mathcal{F}$ runs $\mathcal{U}$ again. According to the general forking lemma, there is a non-negligible probability for $\mathcal{U}$ to produce another qualified signature for the query (m', P^*, Q', c'). $\mathcal{F}$ takes the value e' in the signature as an output about the CDH problem.

Next we analyze the success probability of the forger $\mathcal{F}$ if the forger can finish the game. Suppose the event has a probability ε that $\mathcal{F}$ does not find a match in the R_{list}. In this case, the forger can succeed if the party P^* is just the target for the adversary $\mathcal{U}$ to impersonate, and if $\mathcal{U}$ outputs another qualified transcript for the same $(m^*, P^*, Q^*, e^{*t_{Q^*}^{-1}})$ after the rewinding action. As there are n imitated parties, the probability is $1/n$ that the special P^* is selected as a target. Suppose there are q_e queries to the oracle O_E. Then the probability, according to the general forking lemma, is at least $\epsilon(\epsilon/q_e - 1/q)$ that the adversary $\mathcal{U}$ produces another qualified signature for the same query. So the success probability of the forger $\mathcal{F}$ is $\varepsilon\epsilon/n(\epsilon/q_e - 1/q)$.

The other event has a probability $(1-\varepsilon)$ that $\mathcal{F}$ finds a match in the R_{list}. Since $\mathcal{F}$ runs $\mathcal{U}$ from the beginning, there is another probability $(1-\varepsilon)$ that $\mathcal{F}$ finds a match in R_{list} indexed by $e'^{t_{Q'}^{-1}}$. The party P^* is selected as the target by $\mathcal{U}$ with a probability $1/n$. Suppose there are q_r queries to the oracle O_E^{-1}. The successful rewinding probability is still $\epsilon(\epsilon/q_r - 1/q)$. So the success probability in this case is $(1-\varepsilon)^2\epsilon/n(\epsilon/q_r - 1/q)$.

There is a bad event to make the forger stop the game abnormally. The event is that the P^* is corrupted. Since P^* is selected randomly, the probability to corrupt P^* is $1/n$. The probability of $\mathcal{F}$ to finish the game normally is at least $(1 - 1/n)$.

In summary, the success probability of $\mathcal{F}$ to the SUEF signature is

$$\epsilon_s = \frac{(n-1)\varepsilon\epsilon(q\epsilon - q_e)}{n^2 q q_e} \geq \varepsilon\frac{\epsilon(q\epsilon - q_e)}{2nqq_e}$$

and the success probability to the CDH problem is

$$\epsilon_c = \frac{(n-1)(1-\varepsilon)^2\epsilon(q\epsilon - q_r)}{n^2 q q_r} \geq (1-\varepsilon)^2\frac{\epsilon(q\epsilon - q_r)}{2nqq_r}$$

Finally, we analyze the run time of the forger $\mathcal{F}$. The simulation of O_E^{-1} needs one exponentiation time τ_e for each query. Suppose the existential forgery time of $\mathcal{F}$ is τ_f. When P^* is queried to produce a transcript, there are two times τ_e and one τ_f for each query. Suppose the P^* is queried q_s times. The time is about $t + (q_r + 2q_s)\tau_e + q_s\tau_f$ for $\mathcal{F}$ to wait until the event $\mathcal{B}$ occurs. The rewinding needs about half the above time. So the overall time for $\mathcal{F}$ to break the SUEF signature is about $t_s \approx 1.5(t + (q_r + 2q_s)\tau_e + q_s\tau_f)$. To solve the CDH problem, $\mathcal{F}$ has to run the game again after the event $\mathcal{B}$ occurs. The overall time for $\mathcal{F}$ to solve the CDH problem is about $t_c \approx 2.5(t + (q_r + 2q_s)\tau_e + q_s\tau_f)$. □

Remark 6. It is not new to model a symmetric encryption and decryption as random oracles. This method appeared in [11] when the security of a ring signature was proven.

Remark 7. The general forking lemma is applicable in contexts *other than* standard signature schemes since it only considers the inputs and outputs. We refer our readers to [26] for the detailed reasons.

Proposition 2. *The protocol λ_{DMI} is transcript deniable if the DDH problem is hard.*

Proof. Suppose a simulator S_λ that on input a message m from A to B, produces a transcript as follows:

- Existentially forge a signature δ_F for a random message.
- Randomly select $e_F \in_R \mathbb{G}$.
- Set the simulated transcript as (m, A, e_F, δ_F).

Suppose an adversary $\mathcal{D}$ that claims to distinguish the simulated transcript from a real one without the private key of B and session-state values for the simulated transcript and the real one with a probability ϵ.

Suppose a DDH problem solver S_D which takes as input a DDH problem instance (g, g^x, g^y, g^z). S_D plays with $\mathcal{D}$ using S_λ.

- S_D sets n-imitated parties and runs the function I for them with an exception that the public trapdoor key of a random party B^* is g^y.
- S_D answers queries of $\mathcal{D}$ as follows.
  - When a party is required to send a real message, S_D runs the party according to the protocol specification. When a simulated message is required, S_D runs the S_λ for the party.
  - When a party is required to receive a message, S_D runs according to the protocol specification.
  - When $\mathcal{D}$ corrupts a party, the private key of the party is given to $\mathcal{D}$. If B^* is corrupted, S_D fails.
  - When $\mathcal{D}$ tries to reveal session-state values, S_D gives the values to $\mathcal{D}$.
- S_D produces a challenge message

$$m, A^*, e = g^z, \delta = Sign_{sk_{A^*}}(E_{H(m,A^*,B^*)}(g^x))$$

  using g^x, g^z and the private key of A^*.
- S_D continues to answer queries of $\mathcal{D}$. However, the session-state values about the challenge is not allowed to be revealed. And obviously, the challenge message should not be received by B^*.
- Finally, $\mathcal{D}$ produces a bit $b = 0$ to denote the challenge message is a real transcript, or $b = 1$ otherwise.
- S_D guesses the input tuple is a DDH tuple if $b = 0$ and is not if $b = 1$.

If the input tuple is a DDH tuple, the challenge message is just a qualified transcript. That is, it is a possible real transcript.

Comparatively, if the tuple is not a DDH tuple, the challenge message is not qualified. It is indistinguishable from a simulated one. At first, the signature part is indistinguishable since it is an SUEF signature. Secondly, the value e and e_F are both random values in the group $\mathbb{G}$.

So S_D has the same advantage as $\mathcal{D}$ if the game does not fail. As B^* is selected randomly, the failure probability is $1/n$. So the success probability of S_D is at least $(1 - 1/n)\epsilon \geq \epsilon/2$. □

Proposition 3. *The protocol λ_{DMI} is full deniable.*

Proof. Suppose a simulator $S_{\lambda}^{pri_B}$ that on input a message m from A to B, produces a transcript as follows:

1. Existentially forge a signature δ_F for a random message m_F.
2. Compute $R = E^{-1}_{H(m,A,B)}(m_F)$. If $R \notin \mathbb{G}$, return to step 1. Else compute $e_F = R^{t_B}$.
3. Set the simulated transcript as (m, A, e_F, δ_F).

The values δ_F and δ are indistinguishable, since we use an SUEF signature.

The randomness of e in a real transcript is determined by the effective random value r. That is, the selecting of r will lead to a valid signature. Suppose the number of effective r is denoted by $\#M_r$.

The randomness of e_F is determined by the random message m_F and the re-computing of a forged message-signature pair. Suppose the number of effective m_F is denoted by $\#M_F$. Then when $\#M_F \approx \#M_r$, the simulated transcripts are indistinguishable from real ones.

Remark 8. The numbers $\#M_F$ and $\#M_r$ are related to concrete protocols. A concrete protocol will prove $\#M_F \approx \#M_r$.

5 The Concrete Protocol

The signature scheme of Rabin [24] is a satisfactory SUEF scheme if the hash function is not used. At first, if an adversary can win the attacking game in section 4.1, the adversary can be used to solve the integer factorization problem. So it is a secure SUEF scheme. Secondly, a forged signature has the same distribution as a real one. We set the big primes in the system parameters as the Blum numbers, and require that the real signature is a quadratic residue, which means that it is the principal square root. A forged signature is also a quadratic residue by design. So it has the indistinguishable property.

The concrete protocol is as follows.

– *Setup:* Assume a sender's public key is (N, p, q, g, H) where $N = p_b q_b$ for two big Blum primes p_b and q_b satisfying $|p_b| = |q_b|$, and (p, q, g, H) is the same as the general construction. The sender's private key is (p_b, q_b). An honest receiver's trapdoor key is $t_R \in \mathbb{Z}_q$. The public trapdoor key is $T_R = g^{t_R}$. We require that $|p| = |N| + 1$.

- *Prove:* To send a message m, the sender randomly selects $r \in \mathbb{Z}_q$ and computes $R_p = g^r$. Then it computes $\kappa = H(m, ID_S, ID_R)$. If $R = E_\kappa(R_p)$ is not bigger than N or $R \bmod N$ is not a quadratic residue, another r is selected. Else, the sender calculates $e \leftarrow (T_R)^r$, and $\delta \leftarrow (R)^{1/2} \bmod N$, where δ is the principal square root. Then the sender sends (m, ID_S, e, δ) to the receiver.
- *Verify:* The receiver checks the equation $E_{H(m,ID_S,ID_R)}(e^{t_R^{-1}}) = \delta^2 \bmod N$. If it holds, the receiver accepts and believes that the message source is ID_S, else rejects.
- *Simulate for full deniability:* With a message m and reviver's trapdoor key t_R, a simulator works as follows. The simulator randomly selects $r_x \in_R \mathbb{Z}_N^*$ and computes $\delta_F = r_x^2 \bmod N$, $m_F = \delta_F^2 \bmod N$. If $\delta_F^2 \leq N$, another r_x is selected. If $R_p = E^{-1}_{H(m,ID_S,ID_R)}(m_F + \tau N) \notin \mathbb{G}$, another r_x is selected, where $\tau \in \{0, \ldots, \lfloor p/N \rfloor\}$ is random selected if multiple values are satisfactory. Else $e_F = R_p^{t_R}$. The message simulated is (m, ID_S, e_F, δ_F).
- *Simulate for transcript deniability:* It is the same as the above simulation procedure with an exception that the value e_F is now randomly selected from the group $\mathbb{G}$.

We need to prove that $\#M_r \approx \#M_F$ to satisfy the Proposition 3.

According to the simulation method, m_F is a quadratic residue. Considering the re-selecting, the number of effective m_F is $\#M_F = \omega 1/2(p_b - 1)(q_b - 1)/4 \approx \omega N/8$, where $1 < \omega < 2$. That is about half of the number of quadratic residues in the group $\mathbb{Z}_N$. The coefficient ω is for $p > N$ and the real distributions of quadratic residues in $\mathbb{Z}_N$ and of the group $\mathbb{G}$ in $\mathbb{Z}_p$. The number of effective r, $\#M_r$, is about the number of elements in the set

$$\{r | E_\kappa(g^r) \bmod N \text{ is a quadratic residue}\}.$$

That number is equivalent to the number of elements in the set

$$\{\delta | \delta \in \mathbb{Z}_N \text{ is a principal square root } \wedge E_\kappa^{-1}(\delta^2 \bmod N + \tau N) \in \mathbb{G}\},$$

which is just the total number of δ_F in the simulation algorithm. So $\#M_F \approx \#M_r$.

6 Comparison

We give a table to summarize the analysis results and compare our paradigm with other protocols. In the table, the left column is the literatures. Except the first three rows, all schemes are MAC based. The other three columns are the properties of NIDA protocols. If a protocol is an "MI-Authenticator", it is unforgeable. The meanings of "Transcript Deniability" and "Full Deniability" are defined in the Definitions 3 and 4 in the Section 3. The symbol "Yes" denotes that the protocol in the literature enjoys the property of that column.

Table 1. Analysis Results of NIDA Protocols

	MI-Authenticator	Transcript Deniability	Full Deniability
[9]	Yes	No	Yes
[10]	Yes	No	No
[5]	Yes	No	Yes
[6]	No	Yes	Yes
[8]	No	No	Yes
[12]	No	No	No
[13]	No	No	No
[14]	No	No	No
[15]	No	No	No
[16]	No	Yes	Yes
[17]	No	No	Yes
[18]	No	No	No
[19]	No	No	Yes
[20]	Yes	No	No
[21]	Yes	No	No
[22]	No	No	Yes
Ours	Yes	Yes	Yes

From the table, we observe the following points.

- There are two protocols which enjoy the transcript deniability property.
- There are two MAC based protocols which are MI-authenticators.
- There is no protocol that is an MI-authenticator, and is transcript and full deniable.

The above observations make our scheme unique. It is an MI-authenticator, and enjoys the properties of transcript deniability and full deniability.

7 Conclusion

We describe a formal model for non-interactive deniable authentication protocols, which captures the KCI and SSR attacks. Then we analyze some NIDA protocols in the model, and we show the vulnerabilities of two protocols. Finally, we give a new paradigm to construct a desirable protocol with proofs and a concrete protocol.

Acknowledgment. This work is supported by the National Natural Science Foundation of China (Nos. 60970144, 60803135, 61070168, 61003244, 61103199), Fundamental Research Funds for the Central Universities (Nos. 10lgpy31, 11lgpy71, 11lgzd06, 10lgzd14), Specialized Research Fund for the Doctoral Program of Higher Education for New Teachers (No. 20090171120006), and Beijing Municipal Natural Science Foundation(No. 4112052).

References

1. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero Knowledge. Journal of the ACM 51(6), 851–898 (2004)
2. Borisov, N., Goldberg, I., Brewer, E.: Off-the-record communication, or, why not to use PGP. In: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society (WPES 2004), pp. 77–84. ACM, New York (2004)
3. IPSEC Working Group: Design Rationale for IKEv2. INTERNET-DRAFT, draft-ietf-ipsec-ikev2-rationale-00.txt, `http://tools.ietf.org/html/draft-ietf-ipsec-ikev2-rationale-00`
4. Shao, Z.: Efficient Deniable Authentication Protocol Based On Generalized Elgamal Signature Scheme. Computer Standards & Interfaces 26(5), 449–454 (2004)
5. Wang, B., Song, Z.: A non-interactive deniable authentication scheme based on designated receiver proofs. Information Sciences 179(6), 858–865 (2009)
6. Lee, W., Wu, C., Tsaur, W.: A novel authentication protocol using generalized ElGamal signature scheme. Information Sciences 177(6), 1376–1381 (2007)
7. Chou, J., Chen, Y., Huang, J.: A ID-Based Deniable Authentication Protocol on pairings, `http://eprint.iacr.org/2006/335`
8. Meng, B.: A secure Non-interactive Deniable Authentication Protocol with Strong Deniability Based on Discrete Logarithm Problem and its Application on Internet Voting Protocol. Information Technology Journal 8(3), 302–309 (2009)
9. Lu, R., Cao, Z., Wang, S., Bao, H.: A New ID-Based Deniable Authentication Protocol. Informatica 18(1), 67–78 (2007)
10. Susilo, W., Mu, Y.: Non-interactive Deniable Ring Authentication. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 386–401. Springer, Heidelberg (2004)
11. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
12. Lu, R., Cao, Z.: Non-interactive deniable authentication protocol based on factoring. Computer Standards & Interfaces 27(4), 401–405 (2005)
13. Lu, R., Cao, Z.: A new deniable authentication protocol from bilinear pairings. Applied Mathematics and Computation 168(2), 954–961 (2005)
14. Qian, H., Cao, Z., Wang, L., Xue, Q.S.: Efficient non-interactive deniable authentication protocols. In: The Fifth International Conference on Computer and Information Technology, CIT 2005, pp. 673–679. IEEE (2005)
15. Shi, Y., Li, J.: Identity-based deniable authentication protocol. Electronics Letters 41(5), 241–242 (2005)
16. Brown, D.: Deniable Authentication with RSA and Multicasting, `http://eprint.iacr.org/2005/056`
17. Huang, H., Chang, C.: An efficient deniable authentication protocol. In: International Conference on Cyberworlds, pp. 307–310. IEEE (2005)
18. Fan, C., Zhou, S., Li, F.: An Identity-based Restricted Deniable Authentication Protocol. In: 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, pp. 474–478. IEEE (2009)
19. Wu, T., Zhang, W., Liu, Z., Mu, C.: An efficient deniable authentication protocol. In: International Conference on Management and Service Science, MASS 2009, pp. 1–4 (2009)
20. Xin, X., Chen, D.: A Secure and Efficient Deniable Authentication Protocol. In: WASE International Conference on Information Engineering, ICIE 2009, vol. 2, pp. 172–175 (2009)

21. Xin, X., Chen, D.: ID-based Non-interactive Deniable Authentication Protocol from Pairings. In: International Conference on E-Business and Information System Security, EBISS 2009, pp. 1–4 (2009)
22. Wei, S.: ID-based non-interactive deniable authentication protocol. In: Fifth International Conference on Information Assurance and Security, IAS 2009, pp. 479–482 (2009)
23. Di Raimondo, M., Gennaro, R.: New approaches for deniable authentication. Journal of Cryptology 22(4), 572–615 (2009)
24. Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. MIT Laboratory for Computer Science Technical Report MIT/LCS/TR-212, MIT, MA (1979)
25. Li, G., Xin, X., Li, W.: An Enhanced Deniable Authentication Protocol. In: International Conference on Computational Intelligence and Security, CIS 2008, vol. 1, pp. 336–339. IEEE (2008)
26. Bellare, M., Neven, G.: Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. In: Proceedings of the 13th Association for Computing Machinery (ACM) Conference on Computer and Communications Security (CCS), pp. 390–399. ACM, Alexandria (2006)