An Improvement in Strong Designated Verifier Signatures

Fuw-Yi Yang, Ling-Ren Liang, Chun-Chin Hsu

Computer Science and Information Engineering,

Corresponding Author yangfy@cyut.edu.tw, s9827625@cyut.edu.tw

Industrial Engineering and Management, cchsu@cyut.edu.tw

Chaoyang University of Technology, Wufeng District, Taichung, Taiwan

Abstract

Designated verifier signature schemes are applied in situations such as electronic voting and electronic auctions. Signature recipients can ascertain the contents of the document, but cannot verify the identity of the signer to a third party. This study reviewed the protocols of past research and analyzed whether these protocols would disclose signer identities under various types of attacks. The paper also proposes a strong designated verifier signature scheme that possesses more efficient calculation as well as simpler string encoding for signers. Based on the difficulty of Gap Diffie-Hellman problem and in the Standard Model, the scheme is existentially unforgeable under the adaptively chosen message attacks.

Keywords: deniable authentication, designated verifier signature, signature simulation, unforgeability

1. Introduction

Digital signatures are a technology that provides electronic documents (or messages) with properties of authenticity, non-repudiation, and integrity. That is, the document recipient can verify the signer identity, the signer cannot repudiate the document, and the document has not been tampered with. Therefore, many e-commerce systems are using digital signatures in the hope of increasing customer confidence and avoiding disputes, such as electronic financial transfers, electronic contracts, and business transactions.

In conventional digital signature technology [6, 23], a private key known only to the signer is used to sign documents and generate digital signatures. Other users can verify the validity of the signature with the signer’s public key, ensuring the integrity of the signer’s identity and documents. However, in some circumstances it is not appropriate for the validity of the signature to be verifiable by anyone. For example, the quotation documents that a seller signs to a buyer may become grounds for the buyer to request lower quotes from other sellers; bidding documents signed by bidders may contain prices or other sensitive content that are also revealed if the document is disclosed; in electronic voting, publicly verified votes will cause voters to lose the confidentiality of their political preferences.

In solution to the issues above, cryptographers have proposed designated verifier signatures [12], which use the public keys of both the designated verifier and the signer to sign the document; only the designated verifier can verify the validity of the document. In comparison with traditional digital signatures, designated verifier signatures cannot achieve non-repudiation, but they provide deniable authentication, a technology developed by Deng et al. in 2001 [3]. Deniable authentication possesses
the following two features: 1. only the designated verifier can verify the true source of the signature; and 2. the designated verifier cannot prove the true source of the signature to a third party. In designated verifier signatures, both the signer and the designated verifier can create indistinguishable digital signatures, achieving the two features above.

There are many researches on the schemes of designated verifier signatures or strong designated verifier signatures [7, 9-11, 29]. They are often applied in electronic auctions and voting [15, 28]. In electronic auctions, for example, the bidder (sender) signs a bidding document to the auctioneer (designated recipient). Once the bidder chooses a price and sends the bid, only the auctioneer can verify the true source and integrity of the bidding document, but the auctioneer cannot prove them to other bidders during the course of the auction. This prevents auctioneers from exploiting the auction prices. In 1996, Jakobsson et al. were the first to introduce designated verifier signatures and the applications [12]. However, in the event that an attacker intercepts the signature before the designated recipient receives it, the attacker will be able to verify the signature as well as ascertain the identity of the signer by obtaining the public keys of the signer and the designated recipient. In 2004, Saeednia et al. proposed strong designated verifier signatures [24] to address the aforementioned weaknesses. Combining Schnorr’s signature scheme [25] and Zheng’s signcryption scheme [30], strong designated verifier signatures require the private key of the designated verifier to verify the signature, preventing attackers from detecting the signer’s identity in the event of interception.

However, Lee and Chang [13] indicated that the scheme by Saeednia et al. still had security flaws, as the signer’s private key can verify the signature. If accidentally compromised and obtained by an attacker, the signer’s private key can be intercepted and used to verify the signature, disclosing the signer’s identity. Consequently, Lee and Chang proposed a new strong designated verifier signature scheme founded on Schnorr’s signature scheme and the authenticated encryption scheme of Wang et al. [27]. The new scheme could withstand the attacks mentioned above. In other words, even if an attacker obtained the signer’s private key, they would still face the difficult issue of solving a discrete logarithm problem.

In this study, we first reviewed Lee and Chang’s strong designated verifier signature scheme and indicated the security weaknesses that still exist in it. If the designated verifier’s private key were compromised, an attacker could intercept the signature, verify it, and detect the signer’s identity. In Section 2.4, we explain that the above types of attack seem unstoppable as well as discuss the unfairness and troubles to the signer caused by Lee and Chang’s scheme [13]. After that, we propose a new strong designated verifier signature to improve the efficiencies of the calculation and communication of Saeednia et al.’s scheme [24] and provide deniable authentication as well as the security requirements for strong designated verifier signatures.

2. Review of Lee and Chang’s Scheme

Lee and Chang’s strong designated verifier signature scheme is shown in Figure 1. This section first explains the system parameters and symbols, followed by signature generation, signature verification, a simulation of a
designated verifier generating a signature, and analysis of the reason deniable authentication cannot be achieved.

2.1. System Parameter Settings
- \( p, q; p \) and \( q \) are two large prime numbers generated by the system, satisfying \( q \mid (p - 1) \).
- \( k, k' \): Temporary random numbers chosen by Alice and Bob.
- \( x_A, y_A \): Alice’s long-term private key and public key, where \( y_A = g^{x_A} \mod p \).
- \( x_B, y_B \): Bob’s long-term private key and public key, where \( y_B = g^{x_B} \mod p \).
- \( H(\cdot) \): One-way and collision-free hash function, \( H(\cdot):\{0, 1\}^* \rightarrow \{0, 1\}^l \), \( l \) is the security level parameter, for example, \( l = 160 \).
- \( m \): A document (message)

The hash function \( H(\cdot) \) can be, for example, one of the cryptographic hash functions SHA224, SHA-256, SHA-384, or SHA-512 [16-17], the bit length of hash values are 224, 256, 384, or 512, respectively. The security level \( l = 160 \) is with respect to level 4 security during 2013-2015, as proposed in Lenstra and Verheul [14] and ECRYPT II Recommendations [5]. For 2019, the security level \( l \) will be 192. Other recommendations about security levels can be found in [2,18-19, 22, 26].

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k \in \mathbb{Z}^*_q, r = g^k \mod p )</td>
<td>( m, \sigma = (r, t) )</td>
</tr>
<tr>
<td>( s = k + x_A \cdot r \mod q )</td>
<td>( t = H(m, y_B^s \mod p) )</td>
</tr>
<tr>
<td>( t )</td>
<td>( t\cdot H(m, (r \cdot y_A)^s \mod p) )</td>
</tr>
</tbody>
</table>

**Figure 1.** Lee and Chang’s Strong Designated Verifier Signature Scheme

2.2. Signature Generation Phase
Alice, the document signer, chooses a random number \( k \in \mathbb{Z}^*_q \), and calculates the quantities of \( r, s, t \) as follows:
- \( r = g^k \mod p \)
- \( s = k + x_A \cdot r \mod q \)
- \( t = H(m, y_B^s \mod p) \)

The designated verifier signature is \( \sigma = (r, t) \), and after signing the document, Alice sends the signed signature \((m, \sigma)\) to the designated verifier, Bob.

2.3. Signature Verification Phase
After Bob receives the signed signature \((m, \sigma)\), he can verify the identity of the signer and whether the signature is correct through the following formula:
- \( t\cdot H(m, (r \cdot y_A)^s \mod p) \)

If correct, then Bob is convinced that this document is indeed a valid document signed by Alice.
2.4. Signature Simulation Phase

In this signature simulation phase, Bob can generate a signature \( \sigma \) with the following calculations, which can also pass the verification above. First Bob chooses a random number \( k' \in \mathbb{Z}_q^* \), and calculates \( r', t' \) as follows:

\[
\begin{align*}
  r' &= g^{k'} \mod p \\
  t' &= H(m, (r' \cdot y_A)^{tB} \mod p)
\end{align*}
\]

Using the formulas above, Bob’s simulated signature is \( \sigma' = (r', t') \), which can also be successfully verified.

2.5. Lee and Chang’s Scheme Cannot Achieve Deniable Authentication

We discovered that under normal circumstances, people in general cannot ascertain the signer’s identity with Lee and Chang’s scheme. As Alice’s signature, \( \sigma \), and Bob’s simulated signature, \( \sigma' \), are both randomly distributed in the same space, \((\mathbb{Z}_p^* \times \{0, 1\})^3\), they are indistinguishable.

However, if the private key of the designated verifier, Bob, is unintentionally compromised and obtained by an attacker, then the attacker will be able to verify the signature after intercepting it, and further detect the identity of the signer. This is equivalent to the attack on Saeednia et al.’s scheme [24] by Lee and Chang [13] with the exception of the substitution of the designated verifier’s private key for the signer’s private key. In short, an attacker obtaining the private key of the signer or the designated verifier is a threat to the deniable authentication of Saeednia et al.’s scheme [24]; the compromise of the designated verifier’s private key is also a threat to the deniable authentication of Lee and Chang’s scheme [13] and for the moment, there is no solution.

Lee and Chang’s scheme [13] regulates that only the designated verifier can verify the signature, and not the signer, which doesn’t seem reasonable. For example, a signer sends a signed document \((m, r, t)\), but the designated verifier receives \((m', r, t)\) due to factors such as transmission noise or attacks. If the document \(m'\) is favorable to the designated verifier, they may declare the verification successful without actually carrying it out, causing problems for the signer.

An attacker ascertaining the identity of the document signer as a result of the compromise of the signer’s or designated verifier’s private key may seem inconceivable. However, the concept is similar to full perfect forward secrecy [1] required in key agreement, in which the two parties exchanging session keys, the sender and the recipient, employ the session keys to encrypt messages. If the private keys of both sides are compromised, the session-key-encrypted cipher text is still unlikely to be deciphered. This concept is akin to the purpose of the designated signer’s signature; in the event that the private key of the signer or the designated verifier is compromised, the identity of the signer is still protected.

3. Our Proposed Scheme

The descriptions and analyses above show that Lee and Chang’s scheme [13] is unfair to the signer. This section proposes a new method that not only achieves the functions of Saeednia et al.’s scheme [24] but also has shorter signature length and
smaller amount of calculation. In other words, our new method provides deniable authentication and achieves the security requirements of designated verifier signatures.

The system parameters are almost like that described in the subsection 2.1. The proposed scheme makes slight modifications as follows. One-way hash functions, 
\[ H(\cdot):\{0, 1\}^*\to \mathbb{Z}_q, \quad H_1(\cdot): \mathbb{Z}_q^*\to \mathbb{Z}_p \]
g is an element of \( \mathbb{Z}_p^* \) and generates a group \( G \), i.e. \( G \) is a proper subgroup of \( \mathbb{Z}_p^* \).

**3.1. Signature Generation Phase**

First, Alice the signer calculates the Diffie-Hellman key \( y_{AB} = (y_B)^A \mod p \). Then hashes it under \( H_1(\cdot) \), and chooses a random number \( k \in \mathbb{Z}_q^* \). Using the formula below, Alice calculates \( e \) and the designated verifier’s signature \( \sigma = (e, k) \).

\[
e = H(m, H_1(y_B^A \mod p)^k \mod p)
\]

Once done, Alice sends document \( m \) and signature \( \sigma \) to Bob.

**3.2. Signature Verification Phase**

On receiving the document and signature, Bob verifies whether the document was signed by Alice using the following formula:

\[
e' = H(m, H_1(y_B^{xA} \mod p)^k \mod p)
\]

If correct, Bob is convinced that the document was signed by Alice.

**3.3. Signature Simulation Phase**

In this phase, Bob chooses a random number \( k' \in \mathbb{Z}_q^* \), calculates \( e' \), and simulates signature \( \sigma' \). The formula for \( e' \) is as follows:

\[
e' = H(m, H_1(y_B^{xA} \mod p)^{k'} \mod p)
\]

The signature simulated by Bob is \( \sigma' = (e', k') \), which can pass the verification above. No one else besides Alice and Bob can verify the authenticity of the signature. The proposed designated verifier scheme is shown in Figure 2.

**Figure 2.** The Proposed Designated Verifier Signature Scheme

Lee and Chang’s [13] scheme (Figure 1) requires two exponential computations for both signing the document and verifying the signature; Saeednia et al.’s [24] scheme requires one exponential computation in document signing and two in signature verification. The scheme we propose (Figure 2) only requires one exponential computation each. Suppose a one-way hash function \( H(\cdot) \) outputs a hash string with 160 bits. The bit lengths of \( \mod p \) and \( \mod q \) are 1024 bits and 160 bits, respectively. In Lee and Chang’s scheme [13], the bit length for the signature \( \sigma \) is 1184 bits; in
Saeednia et al.’s scheme [24], the bit length for the signature \( \sigma \) is 480 bits. The bit length for the signature in our scheme is only 320 bits. From this, we can see that the required number of calculations for document signing and signature verification is superior to Lee and Chang’s scheme and the bit length for signature \( \sigma \) is the most concise. Table 1 summarizes the data above.

### Table 1. Comparisons of Computational and Communicational Costs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Signing (MEs)*</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Signature Verification (MEs)</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Length of signature (bits)</td>
<td>1184</td>
<td>480</td>
<td>320</td>
</tr>
</tbody>
</table>

*MEs: computations of modular exponentiations

### 4. Security Analysis

In this section, we will discuss the correctness of the proposed designated verifier signature scheme (in which the signature can be correctly verified). With the exception of the signer and the designated verifier, third parties cannot create signatures within the polynomial time (unforgeability), ascertain the signer’s identity (signer ambiguity), or verify the signature (strong designated verifier signatures). In the event that the private key of the signer or the designated verifier is compromised, other third parties cannot ascertain the signer’s identity (resistance to key compromise attack).

Sub-section 4.2 goes further steps to prove the security of the proposed scheme based on Gap Diffie-Hellman (GDH) assumption, Computational Diffie-Hellman (CDH) assumption and Decisional Diffie-Hellman (DDH) assumption. The following gives a brief description for CDH, DDH, and GDH, for further details, please refer [4, 20-21]. Suppose \( G \) is a multiplicative group, the order of which is a large prime \( q \); \( g \in G \) and \( g \) could generate the multiplicative group \( G \).

The CDH problem is as follows: randomly chooses two elements \( X \) and \( Y \) from \( G \), finds \( Z = g^{xy} \in G \), where \( X = g^x \), \( Y = g^y \in G \) (the quantities of \( x \) and \( y \) are unknown). The CDH assumption implies that CDH problem is hard to solve. Namely, the probability of solving CDH problem is negligible.

The DDH problem is similar as follows: As \( G \) chooses randomly three large numbers \( X \), \( Y \) and \( Z \), to check whether \( z \) is equal to \( xy \) modulo \( q \) ( \( z \equiv xy \mod q \) ), where \( X = g^x \), \( Y = g^y \) and \( Z = g^z \in G \). In the above, the tuple \( (X, Y, Z) \) is called a valid Diffie-Hellman tuple if \( z = xy \mod q \).

The GDH problem states that: Given DDH oracle solve CDH problem, where DDH oracle is a deterministic algorithm answers whether a given tuple \( (X, Y, Z) \) is a valid Diffie-Hellman tuple. Namely, given a pair \( (X, Y) \), a DDH oracle and system parameters \( (g, p, q) \), compute the quantity \( Z = g^{xy} \mod p \). While solving the quantity \( Z = g^{xy} \mod p \), a DDH is available to answer whether \( (X', Y', Z') \) is a valid Diffie-Hellman tuple. The GDH assumption implies that CDH problem (finding \( Z = g^{xy} \mod p \) ) is hard to solve, despite the assistance of DDH oracle. The assumption seems strange, because by asking DDH oracle at most \( q - 2 \)
times we obtain the answer. However, in practical implementation $q$ is as large as $2^{160}$. Group $G$ contains $q$ elements make exhaustive search infeasible.

### 4.1. Correctness

The signature and document signed by Alice is $(m, \sigma)$; Bob can verify the correctness of the signature using the verification, $e^\tau H(m, H_1(y_A^B \mod p)^k \mod p)$.

### 4.2. Unforgeability

For an attacker to forge the designated verifier’s signature on a message, the Diffie-Hellman key, $y_{AB}$, must first be calculated. However, without private keys $x_A$ and $x_B$, the calculation of $y_{AB}$ is not possible, including the calculation of the forged key, $H_1(y_{AB})$. If the attacker attempts to guess $H_1(y_{AB})$ and further calculates Alice and Bob’s Diffie-Hellman key, $y_{AB}$, using an off-line password guessing attack, the attacker still must crack the one-way hash function $H(\cdot)$ and $H_1(\cdot)$ so as to obtain $H_1(y_{AB})^k \mod p$ and $y_{AB}$.

In order to prove the property of security, we need a brief description about the security goal and attack model of signature scheme. The terminology *existentially unforgeable* originally defined in [8] is a common security goal of signature. This means that any adversary should have a negligible probability in forging a valid signature on a new message. The attack model is called adaptive chosen message if a signing oracle is available to adversary while forging a signature. Theorem 1 proves the proposed scheme meets the security requirement of signature. The proof is based on the Gap Diffie-Hellman problem.

**Theorem 1.** The proposed scheme is *existentially unforgeable* under the adaptively chosen message attack and the difficulty of Gap Diffie-Hellman problem.

**Proof.** Let $g$ be a generator of $G$ and $G$ be with prime order $q$. The goal is to solve an instance of CDH problem \{\(g, X = g^i \mod p, Y = g^j \mod p\)\} with the aids of a DDH oracle. Assume that there exists a forger $F$. The followings construct an algorithm $B$ that takes advantage of the forger’s ability to solve the Gap Diffie-Hellman problem.

**Setup system parameters:** $B$ chooses a hash function $H(\cdot) : \{0, 1\}^* \rightarrow Z_q$, $H_1(\cdot): Z_p \rightarrow Z_q$, and simulates the response of $H(\cdot)$ and $H_1(\cdot)$. Let $y_A = X$ and $y_B = Y$. The public system parameters $\text{param} = (H(\cdot), H_1(\cdot), g, p, q)$. $B$ sends $\text{param}$ to the forger $F$.

**Extraction of public key and secret key:** The forger $F$ is allowed to query the extraction oracle for any identity $ID_i$. $B$ acts as extraction oracle and prepares an $ID$-**Table** for a set of $n$ entities. If $ID_i \neq A$ and $ID_i \neq B$, it selects $x_i \in Z_q$ at random, computes $y_i = g^{x_i} \mod p$, and inserts the triple $(ID_i, x_i, y_i)$ to $ID$-**Table**. In the case of $ID_i = A$ or $ID_i = B$, the triple $(ID_A, \perp, X)$ or $(ID_B, \perp, Y)$ is inserted respectively. Then $B$ can respond to the queries of extraction made by forger $F$. The forger will get all the secret key and public key of $ID_i$, i.e. $(x_i, y_i)$. However, if $ID_i = A$ or $ID_i = B$, forger will learn only their public keys, i.e. receives $(\perp, X)$ or $(\perp, Y)$.

**Hash operations:** $B$ manages $H$-**Table** and $H_1$-**Table** to handle hash requests to $H(\cdot)$ and $H_1(\cdot)$. On receiving hash request to $H(\cdot)$ on a pair $(m, V)$, $B$ first looks up it in $H$-**Table**. If the pair’s hash value $e$ has been defined, $B$ returns $e$ to the requester. Otherwise, $B$ randomly selects an element $e$ from the set $Z_q$, sends it back as response, and adds the triple $(e, m, V)$ to $H$-**Table**. Similarly, for hash request to $H_1(\cdot)$ on a
quantity $W \in G$, $\mathcal{B}$ looks up it in $H_1$-Table. If the hash value has already been defined, $\mathcal{B}$ sends it back as response. Otherwise, $\mathcal{B}$ selects a random element $f$ from the set $\mathbb{Z}_p^*$ and responds with the quantity $f$. $\mathcal{B}$ also adds the pair $(f, W)$ to $H_1$-Table. Note that both hash functions $H(\cdot)$ and $H_1(\cdot)$ are simulated like practical hash functions. $\mathcal{B}$ randomly chooses their hash values from the corresponding ranges with no exception (or special case). Thus, the simulations are in the standard model rather than in the Random Oracle Model.

**Signing signature:** Forger $\mathcal{F}$ can query the signing oracle for a message $m$, a document signer $ID_i$ and designated verifier $ID_j$, $i \neq j$. $\mathcal{B}$ acts as signing oracle and responds to the queries as follows. Check whether $ID_i$ and $ID_j$ have been queried for extraction oracle. If either $ID_i$ or $ID_j$ has not yet been queried for extraction oracle, $\mathcal{B}$ queries the extraction oracle for identity $ID_i$ or $ID_j$. Then $\mathcal{B}$ gets the key pairs $(x_i, y_i)$ and $(x_j, y_j)$, which are secret key and public key of $ID_i$ and $ID_j$ respectively, by searching the $ID$ table.

Case 1. $(ID_i = A, ID_j = B)$ or $(ID_i = B, ID_j = A)$: $\mathcal{B}$ aborts and terminates the simulation.

Case 2. $(ID_i = A, ID_j \neq B)$ or $(ID_j = B, ID_i \neq A)$: $\mathcal{B}$ signs the message using the private key of $ID_j$ and returns the signature to $\mathcal{F}$.

Case 3. Otherwise: $\mathcal{B}$ signs the message using the private key of $ID_i$ and returns the signature to $\mathcal{F}$.

**Verifying signature:** Forger $\mathcal{F}$ can query the verification oracle for a signed message $(m, e, k)$, where the signer is $ID_i$ and the designated verifier is $ID_j$, $i \neq j$. $\mathcal{B}$ acts as verification oracle and responds to the queries with either valid or invalid. The details are as follows. Like the procedure of signing signature, it first checks whether $ID_i$ and $ID_j$ have been queried for extraction oracle. If either $ID_i$ or $ID_j$ has not yet been queried for extraction oracle, $\mathcal{B}$ queries the extraction oracle for identity $ID_i$ or $ID_j$. Then $\mathcal{B}$ gets the key pairs $(x_i, y_i)$ and $(x_j, y_j)$ of $ID_i$ and $ID_j$. The verification procedure assumes that before constructing signatures, the corresponding hash queries have been queried.

Case 1. $(ID_i = A, ID_j = B)$ or $(ID_i = B, ID_j = A)$: Since $\mathcal{B}$ knows neither $x_i$ nor $x_j$, therefore it verifies the signed message $(m, e, k)$ as follows. Firstly, $\mathcal{B}$ searches the $H$-Table to locate the triple $(e, m, V)$, computes the quantity $f = V^{e/k} \mod p$. Then searches the $H_1$-Table to find the pair $(f, W)$ and calls the DDH oracle to decide whether $(X, Y, W)$ is a valid Diffie-Hellman tuple. $\mathcal{B}$ forwards the DDH’s answers to the requester ($\mathcal{F}$) as response.

Case 2. $(ID_i = A, ID_j \neq B)$ or $(ID_i = B, ID_j \neq A)$: $\mathcal{B}$ verifies the signed message using $ID_i$’s public key and $ID_j$’s private key. Then $\mathcal{B}$ returns the answer to $\mathcal{F}$.

Case 3. Otherwise: $\mathcal{B}$ verifies the signed message using $ID_i$’s private key and $ID_j$’s public key. Then $\mathcal{B}$ returns the answer to $\mathcal{F}$.

**Output forgery:** Eventually, the forger outputs a forged signature $\sigma^* = (e^*, k^*)$ on message $m^*$ and document signer $ID_i^*$ and designated verifier $ID_j^*$, where $i \neq j$. If $ID_i^* = A$ and $ID_j^* = B$, then $\mathcal{B}$ solves the instance of CDH problem $\{g, X = g^* \mod p, Y = g^y \mod p\}$, namely $g^{xy} = W \mod p$.

**Probability analysis:** Assume that forger $\mathcal{F}$ can counterfeit a signature with non-negligible probability $\varepsilon$ in polynomial time $t$, after adaptively querying all of the
required queries. During the simulation of signing signature, case 1 will abort the simulation. If the forger uniformly chooses the signer and designed verifier from the set of $n$ entities, the probability of abort is $2/(n(n - 1)) \approx 2/n^2$. Also the probability that $ID^*_i = A$ and $ID^*_j = B$ or vice versa is $2/n^2$. Thus in the above simulation, $\mathcal{B}$ is able to solve any instance of CDH with non-negligible probability $\varepsilon' = \varepsilon(1 - 2/n^2)(2/n^2) \approx 2\varepsilon/n^2$, in polynomial time $t' = t + O(q_S + q_V)E$, where $E$ is the time to perform a modular exponentiation, $q_S$ is the number of querying signing oracle, $q_V$ is the number of querying verifying oracle. The conclusion contradicts the difficulty of Gap Diffie-Hellman problem. This completes the proof.

4.3. Signer Ambiguity

The probability of choosing a random signature $(m', \sigma')$ from the signatures that Alice sent to Bob, or $\Pr[(m, \sigma) = (m', \sigma')]$, equals $1/(q - 1)$, as the $k$ in $(e, k)$ is chosen from the set $Z_q^*$. Likewise, the probability of choosing a random signature $(m', \sigma')$ from the signatures that Bob simulated, or $\Pr[(m, \sigma) = (m', \sigma')]$, also equals $1/(q - 1)$. Therefore, it is difficult for anyone to distinguish the true signer from Alice’s signature and Bob’s simulated signature.

4.4. Resistance to Key Compromise Attack

In the scheme we proposed, in the event that the private key of the signer or the designated verifier is compromised, the disclosed signature can be verified as follows:

$$e \cdot H(m, y_A^{k \cdot s_B} \mod p)$$
$$e \cdot H(m, y_B^{k \cdot s_A} \mod p)$$

However, the attacker still cannot ascertain the singer’s identity from the formulas above as the private keys used by the signer and the designated verifier has the same format. In addition, both Alice and Bob can create the Diffie-Hellman key $y_{AB}$, so the attacker cannot ascertain the signer’s identity.

5. Conclusions

In this study, we discovered that past schemes [13, 24] can only provide deniable authentication under general circumstances. In the event an attacker obtains the designated verifer’s private key and further intercepts and verifies the signature, the deniable authentication in the schemes [13, 24] are threatened with no foreseeable solution. Lee and Chang’s scheme [13] limit the power of verification of the signature to the designated verifier, an unfair condition to the document signer. Therefore, this study established a new designated verifier signature scheme, founded not on Lee and Chang’s scheme [13] but on Saeednia et al.’s scheme [24]. The length of the signature was shortened by 33 % and the required amount of calculations by 50 %. At the same time, the security requirements of designated verifier signature were also met. Specifically, the paper reduces the Gap Diffie-Hellman problem to the scheme’s property of unforgeability, in the Standard Model under the attacks of choosing messages adaptively. This would give user confidence to use the proposed scheme in any practical applications.

Acknowledgement
References


