Generic Constructions for Strong Designated Verifier Signature

Deng-Guo Feng*, Jing Xu* and Wei-Dong Chen*

Abstract—A designated verifier signature is a special type of digital signature, which convinces a designated verifier that she has signed a message in such a way that the designated verifier cannot transfer the signature to a third party. A strong designated verifier signature scheme enhances the privacy of the signer such that no one but the designated verifier can verify the signer's signatures. In this paper we present two generic frame works for constructing strong designated verifier signature schemes from any secure ring signature scheme and any deniable one-pass authenticated key exchange protocol, respectively. Compared with similar protocols, the instantiations of our construction achieve improved efficiency.

Keywords—Strong Designated Verifier Signature, Ring Signature, Deniable Authenticated Key Exchange, Provable security

1. INTRODUCTION

The concept of undeniable signature was proposed by Chaum *et al.* [1] to enable signers to have complete control over their signatures. In an undeniable signature scheme, in order to avoid undesirable verifiers getting convinced of the validity of the signature, the verification of the signature requires the participation of the signer in an interactive protocol. However, the signer does not know to whom he is proving the validity of a signature [2, 3]. To solve this problem, Jakobsson *et al.* proposed a designated verifier signature (DVS) scheme [4] in 1996. In a DVS scheme, the signature provides authentication of a message without providing the non-repudiation property of traditional signatures. This is due to the fact that the designated verifier can always construct a signature intended for himself that is indistinguishable from an original signature. Designated verifier signatures have many applications such as in E-voting, call for tenders and software licensing.

However, DVS schemes still have some limitations. In a scenario, where the verifier can prove to a third party that he has not yet received the signature, the third party believes with high probability that the signer has created it. The Strong Designated Verifier Signatures (SDVS), proposed by Jakobsson *et al.* [4], overcome this problem by forcing the designated verifier to use his secret key at the time of verification. Thus, no one else other than the designated verifier can verify an SDVS. This notion was formally defined by Saeednia *et al.* [5] and strengthened

※ This work was supported by the National Grand Fundamental Research (973) Program of China under Grant No. 2007CB311202 and the National Natural Science Foundation of China (NSFC) under Grant No. 60873197
Manuscript received September 20, 2010; accepted November 23, 2010.
Corresponding Author: Jing Xu
* State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing, P.R.China (feng@is.iscas.ac.cn, jingxu0130@gmail.com, zccwd@yahoo.com.cn)

by Laguillaumie *et al.* [6]. Following Saeednia *et al.*'s work, Susilo *et al.* proposed an identity-based SDVS scheme based on pairings in [7].

Jakobsson *et al.* [4] suggested a generic approach for constructing SDVS. In their construction, to sign a message, the signer first uses their signing key to produce a DVS signature for the verifier and then encrypts the signature under the verifier's encryption key. The ciphertext is sent as the SDVS signature to the verifier. This approach is conceptually simple, however, the resulting scheme is not efficient enough, as it requires the encryption of a DVS signature which usually is comprised of several group elements. Nevertheless, the idea of employing certain security protocol as a building block for a generic construction motivates our work.

In this paper, by using a secure ring signature scheme and using deniable one-pass authenticated key exchange protocol as building blocks respectively, we present two secure and generic constructions for SDVS. One merit of this work lies in that we can now design provably secure SDVS schemes in a systematic way by taking advantage of existent research results on ring signature schemes and deniable one-pass authenticated key exchange protocols. In addition, we show that the generic constructions can be instantiated efficiently, and the computation costs of the instantiations are lower than those of similar schemes.

The rest of this paper is organized as follows. Section 2 formally describes a strong designated verifier signature scheme. Section 3 reviews some cryptographic primitives used in our constructions. Our two generic constructions are presented along with the security analysis in Section 4 and Section 5, respectively. The performance comparison of our instantiations with existing DVS schemes is given in Section 6. Section 7 concludes the paper.

2. STRONG DESIGNATED VERIFIER SIGNATURE

Roughly speaking, a strong designated verifier signature (SDVS) allows the signer to sign documents so that only the intended verifier can verify the validity of their signatures and while simultaneously preventing the signature from being arbitrarily disseminated. Below we give the formal definition of an SDVS.

Definition 2.1 A Strong Designated Verifier Signature (SDVS) scheme in the public-key infrastructure setting consists of the following four (probabilistic) polynomial-time algorithms:

- Key Generation: On input 1^k (where k is the security parameter), the algorithm outputs a pair of matching public and secret keys (pk, sk).
- Signature Generation. On input the secret key of the signer S, public keys of S and V (the designated verifier), and the message M, the algorithm outputs a signature σ on M, *i.e.* $\sigma = Sig(sk_S, pk_S, pk_V, M)$.
- Signature Verification. On input the secret key of the verifier V, public keys of S (the signer) and V, the message M and the purported signature σ, the algorithm outputs 1 (*accept*) or 0 (*reject*).
- Simulation. On input the secret key of the verifier V, public keys of S (the signer) and V, and the message M, the algorithm outputs a signature σ on M, *i.e.* $\sigma = Sim(sk_V, pk_S, pk_V, M)$.

The correctness of SDVS requires that $Ver(sk_V, pk_S, pk_V, M, \sigma) = 1$ for all $\sigma = Sig(sk_S, pk_S, pk_V, M)$. Besides the correctness, a secure SDVS should also satisfy three security

requirements: unforgeability, non-transferability and strongness. In the following, we define these properties respectively.

Unforgeability: Unforgeability requires that any third party other than the signer *S* and the designated verifier *V* cannot forge a signature on behalf of *S* with non-negligible probability. Formally, it is defined by the following game played between a game challenger *C* and a probabilistic polynomial-time adversary *A*:

(1) **Setup**: *C* runs the algorithm to obtain the key pairs of *S* and *V*, *i.e.* (pk_S, sk_S) and (pk_V, sk_V), and gives (pk_S, pk_V) to *A*.
(2) **Sign Queries**: *A* can request a signature on a message *M* for the signer *S*, and the designated verifier *V*. In response, *C* outputs a signature σ for a message *M*.
(3) **Verify Queries**: *A* can request a signature verification on a pair (M, σ) for the signer *S*, and the designated verifier *V*. In response, *C* outputs 1 if the signature is correct, and 0 otherwise.
(4) **Output**: Finally, *A* outputs its forgery (M^*, σ^*). It wins the game if M^* has never been queried during the Sign queries and σ^* is a valid signature on M^*.

Definition 2.2: (Unforgeability) An SDVS scheme is said to be $(t, q_{Sig}, q_{Ver}, \varepsilon)$-unforgeable if there is no adversary *A* which runs in time at most *t*, issues at most q_{Sig} Sign queries, and at most q_{Ver} Verify queries, and wins the game with a probability of at least ε.

Non-transferability: Non-transferability means that the signature output by the signer is computationally indistinguishable from the simulated output by the designated verifier. Formally, it is defined by the following game played between a game challenger *C* and a probabilistic polynomial-time distinguisher *D*:

(1) **Setup:** *C* runs the algorithm to obtain the key pairs of *S* and *V*, *i.e.* (pk_S, sk_S) and (pk_V, sk_V), and gives (pk_S, pk_V) to *D*.
(2) **Sign and Verify Queries:** *D* issues queries adaptively for polynomially as many times as in the unforgeability game.
(3) **Challenge:** *D* submits a new message M^* to the challenger *C*. *C* then flips a fair coin $b \xleftarrow{R} \{0,1\}$, generates a signature σ^* and returns it to *D*. If $b = 0$, $\sigma^* = Sig(sk_S, pk_S, pk_V, M^*)$; Otherwise, $\sigma^* = Sim(sk_V, pk_S, pk_V, M^*)$.
(4) **Output:** Finally, *D* outputs a bit *b'* and wins the game if *b* =*b'*.

Definition 2.3 (Non-transferability) An SDVS scheme is said to be non-transferable against an adaptively chosen message distinguisher *D* if $\left|\Pr[b - b'] - 1/2\right|$ is negligible.

Strongness: Strongness requires that anyone who does not know the designated verifier's secret key cannot tell if a signature is from the real signer or other signers. Formally, it is defined by the following game played between a game challenger *C* and a probabilistic polynomial-time distinguisher *D*:

(1) **Setup:** *C* runs the algorithm to generate the key pairs for signers S_1, S_2 and the verifier *V*, *i.e.* (pk_{S_1}, sk_{S_1}), (pk_{S_2}, sk_{S_2}) and (pk_V, sk_V), and gives (pk_{S_1}, pk_{S_2}, pk_V) to *D*.
(2) **Sign and Verify Queries:** *D* issues queries adaptively for polynomially as many times as in the unforgeability game.

(3) **Challenge:** *D* submits a new message M^* to the challenger *C*. *C* then flips a fair coin $b \xleftarrow{R} \{0,1\}$, computes the challenge signature $\sigma^* = Sig(sk_{S_b}, pk_{S_b}, pk_V, M^*)$ and returns σ^* to *D*.

(4) **Output:** Finally, *D* outputs a bit *b'* and wins the game if $b = b'$.

Definition 2.4 (Strongness) An SDVS scheme is said to be strong against an adaptively chosen message distinguisher *D* if $\left|\Pr[b - b'] - 1/2\right|$ is negligible.

3. TOOLS

Our transformation uses several tools, including the ordinary ring signatures defined in subsection 3.1, the deniable authenticated key exchange protocols defined in subsection 3.2, and the public key encryption schemes defined in subsection 3.3.

3.1 Ring Signature

The concept of ring signature was first introduced by Rivest *et al.* in 2001 [8]. In a ring signature scheme, a user wants to sign a message on behalf of a set (or ring) of users which include himself. The verifier can be convinced that the signature was indeed generated by one of the ring members; however, the verifier is unable to tell which member actually produced the signature.

A ring signature scheme consists of the following two algorithms:

Signature Generation: Given a message *M* and the set of ring members $S = \{A_1, A_2, \ldots, A_n\}$, the actual signer $A_s (1 \le s \le n)$ can produce a ring signature σ using the public keys $pk_1, pk_2, \ldots, pk_n$ of the ring and her own private key sk_s.

Signature Verification: Given a message *M* and a ring signature σ, which includes the ring $S = \{A_1, A_2, \ldots, A_n\}$, a verifier can determine whether (M, σ) is a valid ring signature generated by one of the ring members.

The resulting ring signature scheme must satisfy the following security requirements:

Anonymity: Any verifier should not have probability greater than $1/n$ to guess the identity of the real signer who has computed a ring signature on behalf of a ring of n members.

Unforgeability: Any adversary should not have non-negligible probability of success in forging a valid ring signature for some message *M* on behalf of a ring that does not contain himself, even if he knows valid ring signatures for messages, different from *M* that he can adaptively choose.

3.2 Deniable One-Pass Authenticated Key Exchange

The concept of deniable authenticated key exchange (DAKE) was first introduced by Raimondo *et al.* in 2006 [9]. A deniable key exchange allows two parties (sender *A* and responder *B*) to jointly share a secret key while neither of the two nor an outsider can prove to a judge that the communication between the initiator and the responder happened. Another variant of DAKE is a

deniable one-pass authenticated key-exchange protocol in which A sends a single message to B after which both parties share a secret key.

We first recall the security model for authenticated key exchange due to Bellare and Rogaway [10] and then add deniability to this model following the approach in [11].

The model includes a set $\mathcal{U}$ of participant identifiers and each participant $U \in \mathcal{U}$ has a public and private key pair (pk_U, sk_U). The interaction between an adversary A and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. Let U^i denote the instance i of a participant U. All possible oracle queries are listed in the following:

- *Execute* (U^i): This oracle query is used to simulate an eavesdropping attack by the adversary. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.
- *Reveal* (U^i): This query models the misuse of session keys. It returns to the adversary the session key of participant instance U^i, if the latter is defined.
- *Corrupt* (U): The adversary A may request the long-term private key of participant U.
- *Test* (U^i): This oracle query is not used to simulate the adversary's attack, but to define the session key's semantic security. After querying the oracle, the session key of U^i or a random number will be returned according to a predefined random bit b. If $b = 1$, the adversary would learn the session key of U^i; otherwise the adversary only learns a random number with the same length. This query can be called only once.

We say an instance U^i has been accepted if it is successfully completed. An instance U^i is said to be fresh if the following conditions hold: (1) It has been accepted; (2) No *Reveal* or *Corrupt* queries have been made to U or its partner, where the partner denotes the party with which U is presumably interacting.

A secure DAKE protocol should satisfy two security requirements: semantic security of session key and deniability.

Definition 3.1 (Semantic security) Let *Succ (A)* denote the success of A in the Test query. The advantage of A in violating the semantic security of DAKE protocol is defined to be

$$Adv(A) \stackrel{def}{=} 2\Pr[Succ(A)] - 1$$

We say that DAKE protocol satisfies semantic security if the advantage *Adv (A)* is negligible.

Definition 3.2 (Deniability) we say that DAKE protocol satisfies deniability if for any adversary A, for any input of public keys and any auxiliary input, there exists a simulator SIM_A that, running on the same inputs as A, produces a simulated view which is indistinguishable from the real view of A. It is noted that not only the communication during the session but also the value of the session key should be part of the output of the simulation.

3.3 Public Key Encryption

A public key encryption scheme consists of the following polynomial-time algorithms:

- Key Generation: On input 1^k (where k is the security parameter), the algorithm outputs a pair of matching public and secret keys (pk, sk).

- Encryption. On input a public key pk and the message M, the algorithm outputs a cipher text $c = Enc_{pk}(M)$.
- Decryption. On input a secret key sk and a ciphertext c, the algorithm outputs a message M or a special symbol $\perp$ denoting failure.

The first formal security definitions for public key encryption appeared in [12]. The indistinguishability under chosen plaintext attack (IND-CPA) is one of the most natural practical requirements for a public-key cryptosystem. Its intuitive meaning is that a ciphertext does not leak any useful information about the plaintext but its length.

The security is defined by the following game between an adversary and a challenger. The challenger generates a key pair (pk, sk) based on some security parameter k (e.g., a key size in bits), and publishes pk to the adversary. The challenger retains sk. The adversary may perform any number of encryptions or other operations. Eventually, the adversary submits two distinct chosen plaintexts M_0 and M_1 to the challenger. The challenger selects a bit $b \in \{0,1\}$ uniformly at random, and sends the challenge ciphertext $c = Enc_{pk}(M_b)$ back to the adversary. The adversary is free to perform any number of additional computations or encryptions. Finally, it outputs a guess for the value of b. We say the adversary wins if it correctly identifies b.

Definition 3.3 (IND-CPA) A public key encryption scheme is IND-CPA secure if every probabilistic polynomial time adversary wins the above game with a probability of $1/2 + \varepsilon(k)$, where $\varepsilon(k)$ is a negligible function in the security parameter k.

4. GENERIC CONSTRUCTION FROM RING SIGNATURE

We now propose a generic approach to construct a strong designated verifier signature scheme. In our proposal, we employ a secure 2-party ring signature as the building block.

4.1 Construction

Let RS be a 2-party ring signature scheme that is secure as defined in Section 3.1. Then the generic construction proceeds in the following steps:

Key Generation: This phase is the same as the key generation phase of the RS scheme. Suppose that the signer S and the designated verifier V obtain their respective key pairs (pk_S, sk_S) and (pk_V, sk_V).

Signature Generation: Given the key pairs (sk_S, pk_S) of the signer S, the public key pk_v of the designated verifier V and message M, this phase computes the signature σ as follows: S chooses a random number r and computes $Enc_{pk_V}(r)$, $M' = h(M, pk_S, pk_V, r, Enc_{pk_V}(r))$, where h is a hash function and $E_{pk_V}(r)$ means the encryption of r using V's public key pk_v. Then, S starts the signature generation phase of the **RS** scheme and generates the signature $\sigma = (Enc_{pk_V}(r), RS.Sig(M'))$.

Signature Verification: Upon receiving the signature σ on message M, the designated verifier V decrypts $Enc_{pk_V}(r)$ by using his private key sk_v and obtains r. Then, he com-

putes $M'=h(M, pk_S, pk_V, r, E_{pk_V}(r))$, starts the signature verification phase of the **RS** scheme $(RS.Ver(M', \sigma))$, and returns *accept* or *reject*.

Simulation: Given the key pairs (sk_V, pk_V) of the verifier *V*, the public key pk_S of the signer *S* and message *M*, this phase computes the signature σ as follows: *V* chooses a random number *r* and computes $Enc_{pk_V}(r)$, $M'=h(M, pk_S, pk_V, r, Enc_{pk_V}(r))$. Then, *V* starts the signature generation phase of the RS scheme and generates the signature $\sigma = (Enc_{pk_V}(r), RS.Sig(M'))$.

4.2 Security Analysis

We now investigate the security of our generic construction presented above. The analysis concerns the three security properties defined in section 2.

Theorem 4.1: *Our generic construction GC is unforgeable as long as the underlying 2-party ring signature scheme RS satisfies the unforgeability.*

Proof: Let *A* be an adversary without the knowledge of the signer *S*'s or the verifier *V*'s secret keys. We assume that *A* can forge the signer *S*'s signature $\sigma = (Enc_{pk_V}(r), RS.Sig(M'))$ in our general construction **GC**. Then *A* can forge the ring signature $RS.Sig(M')$. Therefore, the unforgeability of our construction reduces to the unforgeability of **RS**.

Theorem 4.2: *Our generic construction is non-transferable as long as the underlying 2-party ring signature scheme RS satisfies the anonymity.*

Proof: Since both the signer *S* and the verifier *V* can compute the $RS.Sig(M')$, and thus can obtain the signature $\sigma = (Enc_{pk_V}(r), RS.Sig(M'))$ on any message *M*, where $M'=h(M, pk_S, pk_V, r, Enc_{pk_V}(r))$. In addition, it is infeasible for any probabilistic polynomial-time distinguisher to tell whether the message *M* was signed by the signer *S* or the designated verifier *V*, which follows easily from the anonymity of RS.

Theorem 4.3: *Our generic construction is strong as long as the underlying public key encryption scheme* $Enc_{pk}(\cdot)$ *is IND-CPA secure.*

Proof: In our generic construction, the verification of validity or invalidity of signatures can only be performed by the designated verifier, since his secret key is involved in the verification. This would guarantee that anyone, who does not know the designated verifier's secret key, cannot verify such a signature, nor distinguish the transcripts from random strings of the same length and distribution. This means that our scheme is inherently strong.

4.3 Discrete Logarithm (DL) Instantiation

We now present a concrete instantiation of transforming a DL ring signature scheme [13] into a strong designated verifier signature scheme. As is the case in all discrete logarithm based schemes, we assume that some common parameters are initially shared between the users: a large prime *p*, a prime factor *q* of *p−1*, a generator $g \in Z_p^*$ of order *q* and a one-way hash function *h* that outputs values in Z_q.

Key Generation: The signer *S* chooses his secret key $x_S \in Z_q$ and publishes the corresponding public key $pk_S = g^{x_S} \bmod p$. The verifier *V* also chooses his secret key $x_V \in Z_q$ and publishes the corresponding public key $pk_V = g^{x_V} \bmod p$.

Signature Generation: When *S* wants to sign a message *M* for *V*, she chooses random values *k, r,* $c_2 \in Z_q^*$, and computes

$$z = pk_V^{c_2} g^k \bmod p$$

$$M' = h(M, pk_S, pk_V, pk_V^r)$$

$$c = h(pk_S, pk_V, M', z)$$

$$c_1 = (c - c_2) \bmod q$$

$$s = (k - x_S c_1) \bmod q$$

The signature is then $\sigma = (g^r, s, c_1, c_2)$.

Signature Verification: Upon receiving the signature $\sigma = (g^r, s, c_1, c_2)$ on message *M*, the designated verifier *V* Computes

$$M' = h(M, pk_S, pk_V, (g^r)^{sk_V})$$

$$z = g^s pk_S^{c_1} pk_V^{c_2} \bmod p$$

$$c = h(pk_S, pk_V, M', z)$$

and outputs *accept* if and only if

$$c_1 + c_2 = c$$

holds. Otherwise, outputs *reject*.

Simulation: The designated verifier *V* can also compute a simulated signature σ', which is indistinguishable from *S*'s signature σ. To do so, *V* chooses random values *k', r',* $c_1' \in Z_q^*$, and computes

$$z' = pk_S^{c_1'} g^{k'} \bmod p$$

$$M' = h(M, pk_S, pk_V, pk_V^{r'})$$

$$c' = h(pk_S, pk_V, M', z')$$

$$c_2' = (c' - c_1') \bmod q$$

$$s' = (k' - x_V c_2') \bmod q$$

The simulated signature is then $\sigma' = (g^{r'}, s', c_1', c_2')$.

4.4 RSA Instantiation

We now present a concrete instantiation of transforming an RSA ring signature scheme [13] into a strong designated verifier signature scheme.

Key Generation: We denote by (N_S , e_S) and (N_V , e_V) the signer *S*'s and the verifier *V*'s public keys, respectively, and by d_S and d_V the respective private keys. Let $h_1 : \{0,1\}^* \to Z_{N_V}$ and $h_2 : \{0,1\}^* \to Z_{N_S}$ be hash functions.

Signature Generation: When *S* wants to sign a message *M* for *V*, she chooses two random values *k*, $c_2 \in Z_{N_V}$, and Computes

$$z = h_1(e_S, e_V, M, k)$$

$$s = h_2(e_S, e_V, M, z + c_2^{e_V} \bmod N_V)$$

$$c_1 = (k^{e_V} - s)^{d_S} \bmod N_S$$

The signature is then $\sigma = (s, c_1, c_2)$.

Signature Verification: Upon receiving the signature $\sigma = (s, c_1, c_2)$ on message *M*, the designated verifier *V* computes

$$k = (s + c_1^{e_S} \bmod N_S)^{d_V} \bmod N_V$$

$$z = h_1(e_S, e_V, M, k)$$

and outputs *accept* if and only if

$$h_2(e_S, e_V, M, z + c_2^{e_V} \bmod N_V) = s$$

holds. Otherwise, outputs *reject*.

Simulation: The designated verifier *V* can also compute a simulated signature σ', which is indistinguishable from *S*'s signature σ . To do so, *V* chooses two random values k' , $c_1' \in Z_{N_S}$, and computes

$$s' = h_2(e_S, e_V, M, k')$$

$$z' = h_1(e_S, e_V, M, (s' + c_1'^{e_S} \bmod N_S)^{d_V} \bmod N_V)$$

$$c_2' = (k' - z')^{d_V} \bmod N_V$$

The simulated signature is then $\sigma' = (s', c_1', c_2')$.

5. Generic Construction from Deniable Authenticated Key Exchange

We now propose another generic approach to construct a strong designated verifier signature scheme from deniable one-pass authenticated key exchange (DOP-AKE) protocols.

5.1 Construction

Let DOP-AKE be a deniable one-pass authenticated key exchange protocol that is secure as defined in Section 3.2. Then the generic construction proceeds in the following steps:

Key Generation: This phase is the same as the key generation phase of the DOP-AKE protocol. Suppose that the signer S and the designated verifier V obtain their respective key pairs (pk_S, sk_S) and (pk_V, sk_V).

Signature Generation: Given the key pairs (pk_S, sk_S) of the signer S, the public key pk_V of the designated verifier V and message M, this phase computes the signature σ as follows: S starts authentication phase in the DOP-AKE protocol and generates authentication message T_S. Then S computes the session key SK and $C = H(pk_S, pk_V, M, SK)$, and outputs the signature $\sigma = (T_S, C)$, where $H(\cdot)$ is a secure hash function.

Signature Verification: Upon receiving the signature $\sigma = (T_S, C)$ on message M, the designated verifier V starts the authentication phase in the DOP-AKE protocol and computes the session key SK by using his private key sk_V. Then, he compares $C' = H(pk_S, pk_V, M, SK)$ with C and outputs *accept* if and only if $C' = C$ holds.

Simulation: Given the key pairs (pk_V, sk_V) of the verifier V, the public key pk_S of the signer S and message M, this phase computes the signature σ as follows: According to the deniability of DOP-AKE protocol, there exists a simulator SIM that, running on the same inputs as V, produces a simulated view T_S and a simulated session key SK. Then, V generates the simulated signature $\sigma = (T_S, C = H(pk_S, pk_V, M, SK))$.

5.2 Security Analysis

We now investigate the security of our generic construction presented above.

Unforgeability: For an adversary A to forge a signature on the message M without the knowledge of the signer S's or the designated verifier V's secret keys, A needs to compute $C = H(pk_S, pk_V, M, SK)$. It is impossible for A to learn SK, provided that DOP-AKE protocol satisfies the semantic security of the session key. Therefore, A cannot forge a legal signature.

Non-transferability: Clearly, the deniability property (Definition 3.2) of DOP-AKE protocol guarantees that the transcript view T_S and the session key SK simulated by the designated verifier V are indistinguishable from those generated by the signer S. This means the simulated signature $\sigma = (T_S, H(pk_S, pk_V, M, SK))$ is also indistinguishable from the real signature.

Strongness: In our generic construction, except for the designated verifier V and the signer S, no third party can check the validity of a signature σ for message M. First of all, according to the semantic security (Definition 3.1) of DOP-AKE protocol, no adversary can reveal the session key SK agreed between S and the receiver (designated verifier) V. Furthermore, without the value of SK the third party cannot check the validity of σ, since the session key SK is involved in the verification. This means that our construction is strong as long as the underlying deniable one-pass authenticated key exchange (DOP-AKE) protocol is semantically secure.

5.3 Instantiation I

Many famous authenticated key exchange protocols such as [14] have their deniable one-pass variants. Following the generic construction, we present a concrete instantiation of transforming a DOP-AKE protocol [14] into a strong designated verifier signature scheme. Let *p, q, g* and hash function $h(\cdot)$ be as defined in Section 4.3.

Key Generation: The signer *S* chooses his secret key $x_S \in Z_q$ and publishes the corresponding public key $pk_S = g^{x_S} \mod p$. The verifier *V* also chooses his secret key $x_V \in Z_q$ and publishes the corresponding public key $pk_S = g^{x_V} \mod p$.

Signature Generation: When *S* wants to sign a message *M* for *V*, she chooses random value $k \in Z_q^*$, and computes

$$T_S = g^k \bmod p$$

$$d = h(T_S, pk_S, pk_V)$$

$$SK = h(pk_V^{k+d \cdot sk_S})$$

$$C = h(pk_S, pk_V, M, SK)$$

The signature is then $\sigma = (T_S, C)$.

Signature Verification: Upon receiving the signature $\sigma = (T_S, C)$ on message *M*, the designated verifier *V* computes

$$d = h(T_S, pk_S, pk_V)$$

$$SK^* = h((T_S \cdot pk_S^d)^{sk_V})$$

and outputs *accept* if and only if

$$C = h(pk_S, pk_V, M, SK^*)$$

holds. Otherwise, outputs *reject*.

Simulation: The designated verifier *V* can also compute a simulated signature σ', which is indistinguishable from *S*'s signature σ. To do so, *V* chooses random value $k' \in Z_q^*$, and computes

$$T_S^{'} = g^{k'} \bmod p$$

$$d^{'} = h(T_S^{'}, pk_S, pk_V)$$

$$SK^{'} = h((T_S^{'} \cdot pk_S^{d'})^{sk_V})$$

$$C^{'} = h(pk_S, pk_V, M, SK^{'})$$

The simulated signature is then $\sigma' = (T_S', C')$.

5.4 Instantiation II

We now apply our construction to another concrete instance, using a DOP-AKE protocol variant derived from a signature with message recovery [15]. Let *p, q, g* and hash function $h(\cdot)$ be as defined in Section 4.3.

Key Generation: The signer *S* chooses his secret key $x_S \in Z_p$ and publishes the corresponding public key $pk_S = g^{x_S} \mod p$. The verifier *V* also chooses his secret key $x_V \in Z_p$ and publishes the corresponding public key $pk_V = g^{x_V} \mod p$.

Signature Generation: When S wants to sign a message *M* for *V*, she chooses two random values $u, k \in Z_q^*$, and computes

$$r = pk_V^u g^{-k} \bmod p$$

$$\tilde{r} = r \bmod q$$

$$s = (k - \tilde{r} sk_S) \bmod q$$

$$SK = h\left(g^u\right)$$

$$C = h(pk_S, pk_V, M, SK)$$

The signature is then $\sigma = (r, s, C)$.

Signature Verification: Upon receiving the signature $\sigma = (r, s, C)$ on message *M*, the designated verifier *V* computes

$$\tilde{r} = r \bmod q$$

$$pk_V^u = g^s pk_S^{\tilde{r}} r$$

$$SK^* = h((pk_V^u)^{sk_V^{-1}})$$

and outputs *accept* if and only if

$$C = h(pk_S, pk_V, M, SK^*)$$

holds. Otherwise, outputs *reject*.

Simulation: The designated verifier *V* can also compute a simulated signature σ', which is indistinguishable from S's signature σ. To do so, *V* chooses two random values $r' \in Z_q^*, s' \in Z_q^*$, and computes

$$\tilde{r}' = r' \bmod q$$

$$U' = g^{s'} pk_S^{\tilde{r}'} r'$$

$$SK' = h((U')^{sk_V^{-1}})$$

$$C' = h(pk_S, pk_V, M, SK').$$

The simulated signature is then $\sigma' = (r', s', C')$.

6. EFFICIENCY EVALUATION

In Table 1, we compare our instantiations (including DL instantiation, RSA instantiation, instantiation I and instantiation II) with Saeednia *et al.*'s [5], based on the length of the signature and the required computational cost, where C_e denotes modular exponentiation operation, C_{pe} denotes public encryption operation, C_{pd} denotes public decryption operation, and " Pre" denotes pre-computed operation. For the comparison of signature length, we set p = 512 bits and q = 160 bits for all discrete logarithm based schemes. In order to have comparable security, we set the RSA modulus to 512 bits in the RSA instantiation.

From the comparison one can see that all our instantiations are much more efficient than the scheme in [5] regarding both signature generation and verification, since the scheme [5] needs to be encrypted using a public key in order to provide strongness.

Table 1. Comparisons of performance

	DL instant.	RSA instant.	instant. I	instant. II	[5]
Length	992	1536	672	832	480
Sign Cost	4Pre	2Pre+ $1C_e$	2Pre	3Pre	1Pre + $1C_{pe}$
Verify Cost	$4C_e$	$3C_e$	$2C_e$	$3C_e$	$3C_e$+$1C_{pd}$

7. CONCLUSION

In this paper, we have proposed two secure and generic approaches to constructing a strong designated verifier scheme, employing a secure ring signature scheme and a deniable one-pass authenticated key exchange protocol as the building blocks, respectively. We have also provided formal security proofs for our construction based on the random oracle model. Moreover, the constructions can be instantiated efficiently, and the computation and communication costs of the instantiations are lower than or comparable to those of similar schemes.

REFERENCES

[1] D. Chaum and H. van Antwerpen, *"Undeniable signatures"*, *CRYPTO89*, LNCS Vol.435, Springer-Verlag, 1989, pp.212-216.

[2] Y. Desmedt and M. Yun, *"Weakness of undeniable signature schemes"*, EUROCRYPT91, LNCS Vol.547, Springer-Verlag, 1991, pp.205-220.

[3] M. Jakobsson, *"Blackmailing using undeniable signatures"*, EUROCRYPT94, LNCS Vol.950, Springer-Verlag, 1996, pp.425-427.

[4] M. Jakobsson, K. Sako, and R. Impagliazzo, *"Designated verifier proofs and their applications"*, EUROCRYPT96, LNCS Vol.1070, Springer-Verlag, 1996, pp.143-154.

[5] S. Saeednia, S. Kremer, and O. Markowitch, *"An efficient strong designated verifier signature scheme"*, ICISC03, LNCS Vol.2971, Springer-Verlag, 2003, pp.40-54.

[6] F. Laguillaumie and D. Vergnaud, *"Designated verifier signatures: Anonymity and efficient construction from any bilinear map"*, SCN04, LNCS Vol.3352, Springer-Verlag, 2004, pp.105-119.

[7] W. Susilo, F. Zhang, and Y. Mu, *"Identity-based strong designated verifier signature schemes"*, *ACISP04*, LNCS Vol.3108, Springer-Verlag, 2004, pp.313-324.
[8] R. Rivest, A. Shamir, and Y. Tauman, *"How to leak a secret"*, *ASIACRYPT01*, LNCS Vol.2248, Springer-Verlag, 2001, pp.552-565.
[9] M.D. Raimondo, R. Gennaro, and H.Krawczyk, *"Deniable authentication and key exchange"*, *ACMCCS06*, New York: ACM Press, 2006, pp.400-409.
[10] M. Bellare and P. Rogaway, *"Entity authentication and key distribution"*, *CRYPTO93*, LNCS Vol.773, Springer-Verlag, 1993, pp.232-249.
[11] S.Jiang and R. Safavi-Naini, *"An efficient deniable key exchange protocol"*, *Financial Cryptography08*, LNCS Vol.5143, Springer-Verlag, 2008, pp.47-52.
[12] S. Goldwasser and S. Micali, *"Probabilistic Encryption"*, Journal of Computer and System Sciences, Vol.28, 1984, pp.270-299.
[13] M. Abe, M. Ohkubo, and K. Suzuki, *"1-out-of-n signatures from a variety of keys"*, *ASIACRYPT 2002*, LNCS Vol.2501, Springer-Verlag, 2002, pp.415-432.
[14] Hugo Krawczyk, *"HMQV: A high-performance secure Diffie-Hellman protocol"*, *CRYPTO 2005*, LNCS Vol.3621, Springer-Verlag, 2005, pp.546-566.
[15] K. Nyberg and R.A. Rueppel, *"Message recovery for signature schemes based on the discrete logarithm problem"*, *EUROCRYPT94*, LNCS Vol.950, Springer-Verlag, 1994, pp.182-193.

Deng-Guo Feng

He is a professor and Ph.D. supervisor with the Institute of Software (IOS), Chinese Academy of Sciences (CAS). He is also a member of the Consultative Committee of National Informatization Specialists and director of the State Key Laboratory of Information Security. In 1995, he received his Ph.D. degree in communication and information systems from Xidian University and began to work as a post-doctorate with the Graduate School of the University of Science and Technology of China. In 1997, he joined IOS and was elected into the project of One Hundred Talents of CAS. Prof. Feng's research interests are in the areas of cryptology and information security.

Jing Xu

received her Ph.D. degree from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences. She is currently an associate research professor with the Institute of Software, Chinese Academy of Sciences. Her research interests include cryptology and information security.

Wei-Dong Chen

received his Ph.D. degree from the Graduate University of Chinese Academy of Sciences. His research interests include network security and security protocol.