Simple and Provable Secure Strong Designated Verifier Signature Schemes

Raylin Tso, Takeshi Okamoto[†], and Eiji Okamoto[‡]

Risk Engineering Major
Graduate School of Systems and Information Engineerin
University of Tsukuba,
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan,
raylin@cipher.risk.tsukuba.ac.jp
{ken[†], okamoto[‡]}@risk.tsukuba.ac.jp

Abstract. In this paper, we introduce a simple strong-designated verifier signature (SDVS) scheme which is much more efficient than previously proposed SDVS schemes. In addition, with only one more parameter published by the signer, this scheme can provide signer's forward security. That is, the consistency of a signature cannot be verified by any third party even if he/she knows a signer's private key. Thus the privacy of a signer's identity is protected independently in each signature, if the designated verifier's private key has not been disclosed. In addition, this scheme can be easily modified to a designated verifier signcryption scheme with virtually no additional cost. We will also show that the proposed scheme is provably secure in the random oracle model.

Key words: BDDH problem, CDH problem, designated verifier signature, privacy of signer's identity, provable security, signer's forward security.

1 Introduction

A *Designated Verifier Signature* (DVS) is intended to allow an entity, Alice, to prove the validity of a signature to a specific verifier, Bob, in such a way that although Bob can check the validity of the signature, he cannot transfer this conviction to other third party. This concept was first introduced by Jakobsson et al. in [6] and was formalized and extended to the notion of *Strong* designated verifier signature (SDVS) scheme by Saeednia et al. in [11]. In a DVS scheme, anyone who intercept the signature can verify the consistency of the signature but cannot distinguish whether it is originated from Alice (signer) or Bob (verifier), because both entities are capable of creating such a signature. On the other hand, in a SDVS scheme, only the designated verifier, Bob, is capable of verifying the consistency and validity of the signature so that the privacy of signer's identity is protected (cf., to encrypt a DVS using Bob's public key is an alternative but not efficient way comparing to a SDVS scheme).

A SDVS scheme is very useful in various situations. For example, it can provide freedom from coercion in electronic voting systems. It also provides a way for a merchant and a customer to negotiate for a best price of a purchase without any third party to verify the validity of the negotiated price.

Most of the SDVS schemes proposed up to now are probabilistic, which use one or more random parameters to mask their signatures. We notice that in some schemes (e.g., [7, 11]), although random parameters have been used, the securities of their schemes still depend on only the Deffie-Hellman key pairs. In other words, random parameters are irrelevant to the securities of their schemes. For the worse, in some schemes (e.g., [7, 11, 13]), if the secret term of a random parameter (e.g., secret term: $r \leftarrow_R Z_q^*$, random parameter $Q \leftarrow rP$, P: a generator of an additional group with prime order q) is disclosed, then the signers' private key will be derived according. Therefore, in these schemes, the secret term of a random parameter picked by a signer should be protected with higher secrecy than that of his/her private key.

On the other hand, Rivest et al. mentioned in [10] that a DVS can be realized by a message authentication code (MAC) computed with a shared secret key, but this requires an initial set-up for the key-agreement procedure. Recently, some *one-way and two-party authenticated key agreement* protocols are proposed ([9] and Scheme II in [8]). Using these protocols with a MAC, SDVS schemes can be easily constructed. Furthermore, these key agreement protocols are quite efficient, which implies that any SDVS scheme without lower communication and computation cost than these key agreement protocols may lose their significance in practical use.

In addition, since a main purpose of a SDVS scheme is to protect the privacy of a signer's identity, the secrecy of previously signed signature should not be affected even if any third party knows the signer's private key. This is called *signer's forward security.* Unfortunately, this property is important but has been less considered up to now.

Our Contribution In this paper, we first propose a deterministic SDVS scheme in which the randomization of a signature depends only on the hashed message and its security depends on the secrecy of the Deffie-Hellman key pair. With this technique we can reduce the communication cost of a signature from (at least) 2 of the previously proposed scheme to only 1 and make a SDVS scheme more efficient. Although the scheme is deterministic, it is provably secure in the random oracle model with unforgeability against existential forgery under adaptive chosen message attack, indistinguishability of signer's identity , and non-transferability of a signature. In addition, with only one more parameter picked and published by the signer, our scheme is easily to become *probabilistic*, and signer's forward security can be provided accordingly. Finally, we will show that our scheme can be easily modified to a designated verifier signcryption scheme with virtually no additional cost, which is important and useful in the case if the message is required to be kept secret from any third party.

Related Work and Paper Organization A SDVS scheme is also named as a *deniable authentication protocol* [1, 4, 12] which originated from the property that a signer can later deny his signature. Scheme proposed in [4] is also a deterministic and Diffie-Hellman key pair based SDVS scheme, but their scheme is more like a MAC and have no concrete security proofs. Our scheme is different with [4] and a MAC mainly in the following aspects.

- Our scheme is easy to become probabilistic and the signer's forward security is provided accordingly.
- Our scheme can be modified into a designated verifier signcryption scheme with virtually no additional cost.
- We have concrete proofs and show that our scheme is secure in the random oracle model.

The rest of this paper is organized as follows. In Section 2, we recall some security assumptions and definitions of SDVS schemes. Section 3 and 4 describe our basic scheme ,modified scheme, and their security proofs. In section 5, we show the performance comparison of our schemes with other previously proposed SDVS or DVS schemes. In Section 6, we show our designated verifiers signcryption scheme and its performance compared with other schemes. Finally, our conclusion is formulated in Section 7.

2 Preliminaries

2.1 Hardness Assumption

The proof of security of our scheme can be reduced to two well-known hardness assumptions, Computational Diffie-Hellman (CDH) and Bilinear Decisional Diffie-Hellman (BDDH) assumptions.

Definition 1. CDH Assumption: Let G be a cyclic group of prime order q, the CDH assumption states that given (g, g^a, g^b) for a randomly picked generator g and random $a, b \in \{1, \cdots, q-1\}$, there exists no polynomial time algorithm which can find an element $C \in G$ such that $C = g^{ab}$ with non-negligible probability.

Definition 2. BDDH Assumption: Let G be a cyclic group of prime order q, and let g be a randomly picked generator of G. The BDDH problem is to distinguish 4-tuples of the form (g^a, g^b, g^c, g^{abc}) and (g^a, g^b, g^c, g^d), where a, b, c, d are random elements of $\{1, \cdots, q-1\}$. We say a BDDH problem is hard if there exists no ponlynomial time algorithm which can distinguish d from abc with non-negligible probability.

2.2 SDVS Model

Definition 3. An SDVS scheme with security parameter k consists of the following algorithms:

- System parameter generation algorithm $SysGen$: It takes 1^k as input and the outputs are the public parameters.

- Key generation algorithm $KeyGen$: It takes the public parameters as input and outputs a public/private key pair (pk_i, sk_i) for each entity U_i in the scheme.
- Signing algorithm $Sign$: It takes a message m, a signer U_i's private key sk_i, a verifier U_j's public key pk_j. The output σ is a SDVS of m.
- Verifying algorithm $Veri$: It takes $< \sigma, m, pk_i, sk_j >$ and the public parameters as inputs, outputs "*accept*" if σ is a valid SDVS of m, otherwise, outputs "$reject$".

A properly formed U_j-designated verifier signature must be accepted by $Veri$.

Definition 4. Security Requirements [7] An SDVS scheme must satisfy the following properties:

- **Unforgeability:** Given a pair of signing keys (pk_i, sk_i) and a pair of verifying keys (pk_j, sk_j), it is computationally infeasible, without the knowledge of the secret key sk_i or sk_j, to produce a valid SDVS σ.
- **Non-transferability:** Given a message m and a SDVS σ of this message, it is (unconditionally) infeasible to distinguish a signer from a designated verifier, even if one knows all secrets. In other words, it is infeasible to determine who from the original signer or the designated verifier performed this signature, even if one knows all secrets.
- **Privacy of signer's identity:** Given a message m and a U_j-designated verifier signature σ of this message, it is computationally infeasible, without the knowledge of the key of U_j or the one of the signer, to determine which pair of signing keys was used to generate σ.

In our probabilistic SDVS scheme, an additional security property is provided.

- **Signer's forward security:** If the private key of a signer A is disclosed, then anyone knowing A's private key can of course impersonate A and forge A's signature. But the secrecy of previously established signature signed by A should not be affected. In other words, any previously established signature signed by A should not become verifiable to any adversary even if he/she knows A's private key. In addition, A's identity should not be derived and verified from the previously signed signatures.

3 Basic Scheme (Deterministic)

In this section, we propose our deterministic SDVS scheme. Let G be an additive group of prime order q in which the CDH problem and the BDDH problem are assumed to be hard. For simplicity, we can think of G as a group of points on an elliptic curve over Z_q.

System parameters generation: A trusted authority (TA) who is trusted by all the entities is responsible for the system parameters generation. On input 1^k to the system parameter generation algorithm $SysGen$, $SysGen$ outputs the following public parameters.

- P: a generator of G with order q.
- h: $G \rightarrow Z_q^*$ a hash function.
- $\mathcal{H}$: $\{0,1\}^* \rightarrow G$ a hash function.

Key generation: TA generates each entity's public/private key pair by $KeyGen$ algorithm. For entities $Alice$ and Bob, their public/pribate key pairs are generated as follows:

- Pick up random $\{a, b\} \stackrel{R}{\leftarrow} Z_q^* \times Z_q^*$, and compute $V_a \leftarrow aP$, $V_b \leftarrow bP$.
- The private/public key pair for participant $Alice$ is (a, V_a), and the private/private key pair for participant Bob is (b, V_b).

Signature generation: When $Alice$ wants to sign a message $m \in \{0,1\}^*$ while designates Bob to be the verifier. $Alice$ executes the $Sign$ algorithm and does the following steps:

- Given $Alice$'s private key a, and Bob's public key V_b, compute aV_b.
- Compute $h(aV_b)$ and $\mathcal{H}(m)$.
- The SDVS for m is $\sigma \leftarrow h(aV_b)\mathcal{H}(m)$.

Verification: With the knowledge that the signature σ is signed by $Alice$, then only Bob can verify the validity of the signature. Bob executes the $Veri$ algorithm and does the following steps:

- Given Bob's private key b, and $Alice$'s public key V_a, compute bV_a and $h(bV_a)$.
- Given the message m, compute $\mathcal{H}(m)$ and $\tilde{\sigma} \leftarrow h(bV_a)\mathcal{H}(m)$.
- Accept σ as a valid signature if and only if $\sigma = \tilde{\sigma}$.

A signature σ will always be accepted by Bob if it is properly formed. The consistency of this scheme is straightforward.

3.1 Security

For digital signatures, the widely accepted notion of security was defined by Goldwasser et. al. in [5] as *existential forgery against adaptive chosen-message attack* (EF-ACMA). For a SDVS scheme, it behaves as not only a signature scheme but also an encryption scheme which encrypts a signer and a verifier's identities. In the following definition, we give the EF-ACMA definition in our SDVS scheme which is a modification of the semantic security against passive adversaries for Identity-Based Encryption described in [2]. In our definition, the signer and verifier's identities as well as the public keys can be randomly selected by an adversary in the challenge phase. Although some restrictions on these identities are required, we emphasize the difference that in other SDVS schemes (if they have concrete security proofs), the identities (as well as public keys) of a signer and a verifier in their EF-ACMA proofs cannot be randomly selected by an adversary and was decided and provided by the challenger.

Definition 5. Existential Forgery under Adaptive Chosen Message Attack (EF-ACMA)

Consider the following game played by an adversary $\mathcal{A}$.

1. **Setup.** In the setup phase, the challenger generates all the public parameters and gives them to the adversary.
2. **Phase** 1. The adversary $\mathcal{A}$ is allowed to make the following queries.
   - **$\mathcal{H}$-query:** At any time, $\mathcal{A}$ can ask the $\mathcal{H}$-hash query of a message m_i at his choice.
   - **PrivateKey-query:** At any time, $\mathcal{A}$ is allowed to ask to the private key revealing oracle Pri. The adversary can repeat this multiple times for different public keys.
   - **h-query:** At any time, $\mathcal{A}$ is also allowed to make h-query by given one parameter $Q_i \in G$.
   - **Signing query:** At any time, $\mathcal{A}$ can ask a signing query to the signing oracle Σ by providing a message m_i and two public keys $pk_l, pk_{l'}$. We assume that $\mathcal{A}$ always requests the $\mathcal{H}$-query of a message m_i before it requests a signing query of m_i and it always requests the $\mathcal{H}$-query of the message m^* that it outputs as its forgery. It is trivial to modify any forger to have this property.
3. **Challenge.** The adversary $\mathcal{A}$ submits two public keys pk_{j_1}, pk_{j_2} which he will use to forge a designated verifier signature. These public keys are restricted to the entities U_{j_1}, U_{j_2} that their private keys have not been requested, in addition, any U_{j_2}-designated verifier signature signed by U_{j_1} or U_{j_1}-designated verifier signature signed by U_{j_2} has not been requested to the signing oracle in the previous phase.
4. **Phase** 2. Phase 1 is repeated with the restriction that the adversary $\mathcal{A}$ cannot request the private keys for pk_{j_1} and pk_{j_2}. In this phase, signing queries of signatures between U_{j_1} and U_{j_2} are available.
5. **Forgery.** The adversary submits a forged signature σ^* of a message m^*, where U_{j_1} is the signer and U_{j_2} is the designated verifier. In addition, σ^* is a new signature, which means that the signing query of m^* between U_{j_1} and U_{j_2} has not been requested.

Definition 6. Given a security parameter k, the advantage of a forgery algorithm $\mathcal{A}$ in existentially forging a SDVS of our basic scheme (BS), where $\mathcal{A}$ can access to a signing oracle Σ, a private key revealing oracle Pri and two hash functions $h : G \rightarrow Z_q^*$ and $\mathcal{H} : \{0,1\}^* \rightarrow G$ with restrictions defined in Definition 5, is defined as

$$Adv_{BS,\mathcal{A}}^{EF-ACMA} \triangleq$$

$$Pr\left[Veri(pk_{j_1}, sk_{j_2}, m^*, \sigma^*) = accept \;\middle|\; \begin{array}{l} (P,G) \xleftarrow{R} SysGen(1^k) \\ (sk_i, pk_i, 1 \le i \le n) \xleftarrow{R} KenGen(P, \\ \hfill G, U_i, 1 \le i \le n) \\ (m^*, \sigma^*) \xleftarrow{R} \mathcal{A}^{\Sigma, Pri, h, \mathcal{H}}(pk_{j_1}, pk_{j_2}) \end{array} \right].$$

Here n is the number of entities in the scheme and (pk_{j_1}, pk_{j_2}) are two public keys with restrictions defined in Definition 5.

Definition 7. A SDVS scheme is $(\mathcal{T}, q_h, q_{\mathcal{H}}, q_{Pri}, q_{\Sigma}, \epsilon)$-secure against EF-ACMA if all $\mathcal{T}$-time adversaries making at most q_h h-query, $q_{\mathcal{H}}$ $\mathcal{H}$-query, q_{Pri} PrivateKey-query and q_{Σ} Signing query, respectively, have an advantage ϵ in breaking our scheme.

In the following theorem, we prove that our scheme is secure against EF-ACMA in the random oracle model.

Theorem 1. Unforgeability: Suppose there exists an adversary $\mathcal{A}$ which can $(\mathcal{T}, q_h, q_{\mathcal{H}}, q_{Pri}, q_{\Sigma}, \epsilon)$-break our SDVS scheme via EF-ACMA, then we can construct an algorithm $\mathcal{F}$ which can $(\mathcal{T}', \epsilon')$-break the CDH problem on G where

$$\mathcal{T}' = \mathcal{T} + q_h O(1)_h + q_{\mathcal{H}} O(1)_{\mathcal{H}} + q_{Pri} O(1)_{Pri} + q_{\Sigma} O(1)_{\Sigma} \quad \text{and}$$
$$\epsilon' \geq 1/q_{\Sigma} \cdot (1 - 1/(q_{\Sigma}+1))^{(q_{\Sigma}+1)} \epsilon.$$

Here $O(1)_{\wp}$ with $\wp \in \{h, \mathcal{H}, Pri, \Sigma\}$ is the time unit for replying the corresponding query.

Proof: We show how a CDH problem in G can be solved if a signature of our scheme can be forged.
Initial Stage. $\mathcal{F}$ is given a challenge $(Q, \alpha Q, \beta Q)$ from an outsider where Q is randomly picked from the additional group G, and $(\alpha, \beta) \xleftarrow{R} Z_q^* \times Z_q^*$. $\mathcal{F}$'s goal is to find $\alpha\beta Q \in G$.
Setup. In the setup phase, $\mathcal{F}$ generates a primitive element P of G. On input $(P, G, U_i, 1 \leq i \leq n)$, where $U_i, 1 \leq i \leq n$ denotes the n entities of the scheme, $\mathcal{F}$ runs $KenGen$ and the outputs are the public/private key pairs (pk_i, sk_i) of entity U_i for $1 \leq i \leq n$. At the end of this phase, $\mathcal{F}$ gives all the public parameters $(P, G, pk_i, 1 \leq i \leq n)$ to $\mathcal{A}$ and allows $\mathcal{A}$ to run.

Phase 1. The following queries are available to $\mathcal{A}$ and are controlled by $\mathcal{F}$. Any of its queries may depend on previous answers

- **$\mathcal{H}$-query:** Any time when $\mathcal{A}$ asks the $\mathcal{H}$-hash query of a message m_i. $\mathcal{F}$ picks $r_i \xleftarrow{R} Z_q^*$ and outputs $r_i Q \in G$ as a reply. Each of these queries is recorded in the $\mathcal{H}$-List in the form $(m_i, r_i, r_i Q)$ by $\mathcal{F}$ so as to make sure that each query of different m_i has distinct answer.
- **PrivateKey-query:** Each time when $\mathcal{A}$ provides a public key $pk_i (= x_i P)$ to Pri, $\mathcal{F}$ responds this query with x_i.
- **h-query:** When $\mathcal{A}$ makes a h-query by given a parameter $Q_i \in G$, $\mathcal{F}$ replies with $\hat{\gamma_i} \xleftarrow{R} Z_q^*$. Each of this query/reply pair is recorded in the h-List by $\mathcal{F}$.
- **Signing query:**
  For any signing query σ_i of a message m_i and two public keys $pk_{sign}(= xP), pk_{veri}(= yP)$ provided by $\mathcal{A}$, if $xyP = Q_j$ for some Q_j where $h(Q_j) = \hat{\gamma_j}$

has been requested previously, then $\mathcal{F}$ replies $\sigma_i \leftarrow \hat{\gamma}_j r_i Q$ where $r_i Q = \mathcal{H}(m)$ is extracted from the $\mathcal{H}$-List. Otherwise, $\mathcal{F}$ outputs $\sigma_i \leftarrow \hat{\gamma}' r_i Q$ with $\hat{\gamma}' \xleftarrow{R} Z_q^*$ and records σ_i in the Σ-List and $\hat{\gamma}'$ in the h-List as the result of $h(xyP)$.

Challenge. The adversary $\mathcal{A}$ submits two public keys pk_{j_1}, pk_{j_2} which he will use to forge a designated verifier signature between the two entities U_{j_1}, U_{j_2}. For convenience, we set U_{j_1} as U_A, U_{j_2} as U_B, and $pk_{j_1} = V_a = aP$, $pk_{j_2} = V_b = bP$. These public keys are restricted to the entities that their private keys have not been requested, in addition, any U_B-designated verifier signature signed by U_A or U_A-designated verifier signature signed by U_B has not been requested to the signing oracle in Phase 1.

Phase 2. If $abP = Q_i$ for some $Q_i \in G$ where $h(Q_i)$ has been asked in Phase 1, then $\mathcal{F}$ aborts the challenge and outputs "failure". Otherwise, $\mathcal{F}$ allows $\mathcal{A}$ to repeat Phase 1 with the restriction that $\mathcal{A}$ cannot request the private keys for V_A and V_B. In this phase, $\mathcal{A}$ can initiate the signing query for a signature of a message m signed and designated between U_A and U_B. $\mathcal{F}$ replies with $r\alpha Q (= \alpha r Q)$ where rQ is the reply of $\mathcal{H}(m)$ which $\mathcal{A}$ requested previously, therefore r can be extracted from the $\mathcal{H}$-List.

On the other hand, since a signature can be derived without a signing query if the corresponding $\mathcal{H}$-query, PrivateKey-query and h-query have been asked, we may assume that, in Phase 2, $\mathcal{A}$ requests a signing query to the signing oracle Σ only for a signature signed and verified between U_A and U_B. For any other signature not signed and verified between U_A and U_B, $\mathcal{A}$ use the alternative way (i.e., $\mathcal{H}$-query, PrivateKey-query and then h-query) to find its value. It is trivial to modify any adversary $\mathcal{A}$ to have this property.

Forgery. After q_h h-query, $q_{\mathcal{H}}$ $\mathcal{H}$-query, q_{Pri} PrivateKey-query and q_{Σ} Signing query, the adversary submits a forged signature σ^* of a message m^* signed by U_A and is designated to U_B.

From the above description, anyone can see easily that all the oracles simulated by $\mathcal{F}$ are indistinguishable from real oracles. So everything should go well without any problem if both $\mathcal{F}$ and $\mathcal{A}$ are behaved well. In addition, because of the restrictions described in the Challenge phase, we can conclude that a forged signature between U_A and U_B cannot be launched successfully without the process of Phase 2 (because the value of $h(abP)$ has not been distributed yet before Phase 2). In the following proof, we focus only on the h-query in both Phases and signing query in Phase 2. We assume that $abP \neq Q_i$ for some $Q_i \in G$ where $h(Q_i)$ has been asked in Phase 1. In addition, $\mathcal{A}$ cannot request the singing query (with entities U_A and U_B) of the message m^* in Phase 2 but $\mathcal{H}(m^*)$ must have been queried before it outputs its forgery. We construct a series of games and modify $\mathcal{F}$ in each game. The final variant of $\mathcal{F}$ in the final game thus is the one we want for solving the CDH problem. This idea is inspired by [3].

- <u>*Game 0 :*</u> $\mathcal{F}$ behalves as previous description but records a new list (called $\mathcal{N}$-List) containing a list of tuples $\langle m_i, Q_i, \sigma_i, s_i \rangle$ as explained below. All of these tuples are initially empty. m_i is message of the i-th query for $\mathcal{H}(m_i)$, $Q_i = r_iQ = \mathcal{H}(m_i)$, $\sigma_i = h(abP)\mathcal{H}(m_i) = r_i\alpha Q$ if it has been queried, otherwise σ_i is still empty. s_i maintains empty which will be used from the next game. Finally, in the Forgery phase, $\mathcal{A}$ outputs a forged signature (m^*, σ^*). If σ^* is a valid DVS of m^*, and $m^* = m_{i^*}$ for some i^* where its signing query of the two entities U_A, U_B has not been queried, then $\mathcal{F}$ outputs "success"; otherwise, it outputs "failure". Since the modification in this game will not affect the behavior of $\mathcal{A}$, by Definition 5 and 6, we have

$$\begin{aligned} Adv_{\mathcal{F}}^{Game\ 0} &= Pr\left[\mathcal{F}^{\mathcal{A}}(G, Q, \alpha Q, \beta Q) = success \middle| \begin{array}{l} Q \xleftarrow{R} G \\ \alpha, \beta \xleftarrow{R} Z_q^* \end{array}\right] \\ &= Adv_{BS,\mathcal{A}}^{EF-ACMA} = \epsilon. \end{aligned}$$

- Game 1 : $\mathcal{F}$ behaves as that in *Game* 0 with a difference that, before it replies to $\mathcal{A}$ for $\mathcal{H}(m_i)$ of a message m_i, $\mathcal{F}$ replace s_i in $\mathcal{N}$-List with an element in $\{0,1\}$ by flipping a random cycle. The cycle outputs $s_i \leftarrow 1$ with probability $1/(q_\Sigma + 1)$ and $s_i \leftarrow 0$ with probability $1 - 1/(q_\Sigma + 1)$. Finally, $\mathcal{F}$ outputs "success" if $\mathcal{A}$ succeeds in outputting a forgery (m^*, σ^*) and $s_{i^*} = 1$ for the message m^*. The change in this game will not affect the behavior of $\mathcal{A}$ since $\mathcal{A}$ have no information about any s_i. Thus we have $Adv_{\mathcal{F}}^{Game\ 1} = Adv_{\mathcal{F}}^{Game\ 0} \cdot Pr[s_{i^*} = 1] = \epsilon/(q_\Sigma + 1)$. We define $s_i \in \{0,1\}$ with different probabilities in order to let $\mathcal{F}$ of the following games to have advantages of maximum lower-bound.
- Game 2 In this game, $\mathcal{F}$ functions as that in *Game* 1 but outputs "success" only if $s_{i^*} = 1$ of the message m^* and $s_i = 0$ of the other messages m_i. The same as that in *Game* 1, $\mathcal{A}$ cannot get any information about s_i, so its behavior is independent of any s_i. Sine $\mathcal{A}$ makes at most q_Σ signing queries in Phase 2, and for each signing query of a message m_i in Phase 2, the probability that $s_i = 0$ is $1 - 1/(q_\Sigma + 1)$, therefore, we have $Adv_{\mathcal{F}}^{Game\ 2} \geq Adv_{\mathcal{F}}^{Game\ 1} \cdot Prob[s_{i_j} = 0, 1 \leq j \leq q_\Sigma] = \epsilon/(q_\Sigma + 1) \cdot (1 - 1/(q_\Sigma + 1))^{q_\Sigma} = 1/q_\Sigma \cdot (1 - 1/(q_\Sigma + 1))^{(q_\Sigma + 1)}\epsilon$.
- Game 3 : In this game, $\mathcal{F}$ functions as that in *Game* 2 with the difference that if $\mathcal{A}$ requests a signature on a message m_i for which $s_i = 1$, then $\mathcal{F}$ declares failure and halts immediately. If, finally, $\mathcal{A}$ creates a valid forgery (m^*, σ^*) and $\mathcal{F}$ outputs "success" in *Game* 3, then there is no difference between *Game* 2 and *Game* 3. Therefore, $Adv_{\mathcal{F}}^{Game\ 3} = Adv_{\mathcal{F}}^{Game\ 2} \geq 1/q_\Sigma \cdot (1 - 1/(q_\Sigma + 1))^{(q_\Sigma + 1)}\epsilon$. *Game* 3 provides a shortcut for the case when $\mathcal{F}$'s output is "failure".
- Game 4 : In *Game* 4, if $s_i = 1$ for some m_i, then $\mathcal{F}$ sets $Q_i \leftarrow r_i\beta Q$ in $\mathcal{N}$-List. But no change will be occurred in a query if $s_i = 0$. Since r_i is randomly picked in Z_q^*, r_iQ and $r_i\beta Q$ are both uniform distributions in G. Therefore, this modification is still indistinguishable from a real oracle and $\mathcal{A}$ will behave under $\mathcal{F}$ exactly as it does in previous games. So we have $Adv_{\mathcal{F}}^{Game\ 4} = Adv_{\mathcal{F}}^{Game\ 3} \geq 1/q_\Sigma \cdot (1 - 1/(q_\Sigma + 1))^{(q_\Sigma + 1)}\epsilon$.
- Game 5 : In this final game, whenever $\mathcal{F}$ in *Game* 4 outputs "success", it also outputs "success" in *Game* 5 and, in addition, outputs $(r^*)^{-1} \cdot \sigma^*$,

where m^* is the message for which $\mathcal{A}$ outputs a forged signature σ^*. Clearly, $Adv_{\mathcal{F}}^{Game\ 5} = Adv_{\mathcal{F}}^{Game\ 4} \geq 1/q_{\Sigma} \cdot (1 - 1/(q_{\Sigma}+1))^{(q_{\Sigma}+1)}\epsilon$.

In *Game* 5, if the forgery (m^*, σ^*) $\mathcal{A}$ made is a valid message/signature pair, then $\sigma^* = h(abP)\mathcal{H}(m^*) = \alpha\mathcal{H}(m^*) = \alpha r^* \beta Q$ Consequently, we have $(r^*)^{-1} \cdot \sigma^* = \alpha\beta Q$.

The running time required by $\mathcal{F}$ is the same as $\mathcal{A}$'s running time plus the time it takes to respond to all the queries $\mathcal{A}$ made. Thus, if $\mathcal{A}$ runs in time $\mathcal{T}$, then

$$\mathcal{T}' = \mathcal{T} + q_h O(1)_h + q_{\mathcal{H}} O(1)_{\mathcal{H}} + q_{Pri} O(1)_{Pri} + q_{\Sigma} O(1)_{\Sigma}.$$

□

In the next theorem, we prove that our basic scheme possesses the indistinguishability of signer's identity based on the BDDH problem.

Theorem 2. Privacy of signer's identity: If there exists an algorithm $\mathcal{A}$ which can $(t, q_{\mathcal{H}}, q_{\Sigma}, \epsilon)$-break the indistinguishability of signer's identity, then there exists an algorithm $\mathcal{F}'$ which can $(t + q_{\mathcal{H}}O(1)_{\mathcal{H}} + q_{\Sigma}O(1)_{\Sigma}, \epsilon)$-break the $BDDH$ Problem in G. Here h-query and PrivateKey query are omitted since they are not controlled by $\mathcal{F}'$ in the following proof.

Proof: $\mathcal{F}'$ is given all the elements in G by an outsider $\mathcal{L}$. In particular, P is the generator of G with order q. At any time, $\mathcal{F}'$ can access a reveal oracle Rev (controlled by $\mathcal{L}$) by picking up three elements $< X(= xP), Y(= yP), Z(= zP) > \in_R G^3$ and Rev responds with $xyzP$. We assume the outsider $\mathcal{L}$ (as well as Rev) are well-behaved so Rev always responds $< X, Y, Z >$ with correct $xyzP$. At the end, $\mathcal{F}'$ requests its BDDH challenge by providing two three-tuple $< A_0(= a_0P), B(= bP), C(= cP) > \in G^3$ and $< A_1(= a_1P), B(= bP), C(= cP) > \in G^3$ at his choice to $\mathcal{L}$. Note that $\mathcal{F}'$ does not know the values of $< a_0, a_1, b, c >$. This time, $\mathcal{L}$ outputs only one solution a^*bcP, $a^* \in \{a_0, a_1\}$, of the BDDH problem, and $\mathcal{F}'$ has to distinguish if a^*bcP is the solution of the BDDH problem of $< A_0, B, C >$ or $< A_1, B, C >$.

On the other hand, $\mathcal{F}'$ simulates a SDVS scheme with n entities (for convenience, we denote $\aleph = \{U_1, \cdots, U_n\}$ as the set of these entities) and provides to $\mathcal{A}$ the public key of each entity $U_i \in \aleph$ by randomly picking an element $Q_i \in G$. Denote $PK_{\aleph}$ the set of these public keys. In addition, $\mathcal{A}$ is allowed to make at most $q_{\mathcal{H}}$ $\mathcal{H}$-queries and q_{Σ} signing queries, respectively. $\mathcal{F}'$ is responsible for these queries so $\mathcal{F}'$ has to simulate the random oracle $\mathcal{H}(\cdot)$ and the signing oracle $\Sigma(\cdot)$ well. The same as Theorem 1, we assume that $\mathcal{A}$ always requests the hash of a message m before it requests a signature of m. At the end, $\mathcal{A}$ requests its full-anonymity challenge by providing to $\mathcal{F}'$

1. two signers $U_{\Omega_0}, U_{\Omega_1}$ with public key $V_{\Omega_0}(= \omega_0 P)$, $V_{\Omega_1}(= \omega_1 P)$, respectively,
2. one designated verifier U_{Γ} with public key $V_{\Gamma}(= \gamma P)$, and
3. a message $M \in \{0, 1\}^*$

at his choice where the hash query $\mathcal{H}(M)$ must have been asked but its signing query must not have been asked previously. This time $\mathcal{F}'$ outputs a signature σ^* singed by U_{Ω^*} and $\mathcal{A}$ has to distinguish if $\Omega^* = \Omega_0$ or Ω_1.

We show how B can solve his challenge of a BDDH problem by utilizing $\mathcal{A}$. At any time, when $\mathcal{A}$ asks the hash oracle $\mathcal{H}(\cdot)$ for a message m_i, $\mathcal{F}'$ responds the query by randomly picking an element $R_i \leftarrow G \setminus PK_\aleph$. $\mathcal{F}'$ has also to record the pair (m_i, R_i) in his $\mathcal{H}$-List so as to make sure that each query has distinct answer. In other words, $\mathcal{F}'$ has to make sure that $R_i \neq R_{i'}$ if $m_i \neq m_{i'}$. It is obvious that $\mathcal{H}$ is a random oracle if $q >> n$. Further, each time when $\mathcal{A}$ asks a signature σ_j by providing a signer U_{X_j} (with public key $V_{X_j}(= x_j P)$, a verifier U_{Y_j} (with public key $V_{Y_j}(= y_j P)$ and a message m_j where $\mathcal{H}(m_j) = R_j \leftarrow G \setminus PK_\aleph$, then $\mathcal{F}'$ requests to the reveal oracle *Rev* by providing $< V_{X_j}, V_{Y_j}, R_j >$. The reply from *Rev*, which is $\mathcal{E}_j(= x_j y_j r_j P)$ is assigned by $\mathcal{F}'$ as the reply of σ_j. Since each σ_j is uniformly distributed in G and R_j is randomly picked from $G \setminus PK_\aleph$, so $\mathcal{F}'$ simulates the signing oracle indistinguishably from a real oracle. Finally, after $q_\mathcal{H}$ hash queries and q_Σ signing queries, $\mathcal{A}$ requests its challenge by providing $< U_{\Omega_0}, U_\Gamma, M >$ and $< U_{\Omega_1}, U_\Gamma, M >$ to $\mathcal{F}'$ where $\mathcal{H}(M) = R_M(= r_M P)$ has already been requested. To respond the query, $\mathcal{F}'$ requests its BDDH challenge by providing two triples $< V_{\Omega_0}, V_\Gamma, R_M >$ and $< V_{\Omega_1}, V_\Gamma, R_M >$ to $\mathcal{L}$, where V_{Ω_i} is the public key of U_{Ω_i}, $\Omega_i \in \{\Omega_0, \Omega_1\}$, V_Γ is the public key of U_Γ. The output $\mathcal{E}^*(= \omega^* \gamma r_M P)$ from $\mathcal{L}$, $\omega^* \in \{\omega_0, \omega_1\}$, is assigned by B as the output of $\mathcal{A}$'s challenge. Then, when $\mathcal{A}$ returns an entity U_{Ω_s} with $s = 0$ or $s = 1$ as the answer of his challenge, $\mathcal{F}'$ also returns $< V_{\Omega_s}, V_\Gamma, R_M >$ as the answer of his BDDH challenge. Consequently, if $\mathcal{A}$ breaks the privacy of signer's identity of the proposed scheme with advantage ϵ, then $\mathcal{F}'$ breaks the BDDH problem with the same advantage.

Obviously, $\mathcal{F}'$'s running time exceeds $\mathcal{A}$'s by the amount it takes to answer $\mathcal{A}$'s hash queries and signing queries. So, if $\mathcal{A}$ runs in time t, then $\mathcal{F}'$ runs in time $t + q_\mathcal{H} O(1)_\mathcal{H} + q_\Sigma O(1)_\Sigma$. This ends the proof. $\square$.

Theorem 3. Non-transferability: The proposed scheme provides non - transferability of a signature.

This is straightforward since the operation of a signature and a verifier is done symmetrically. Since both entities (signer and verifier) are capable of creating this signature, no any third party can distinguish a signer from a designated verifier even if he knows all secrets. In other words, no any third party can determine who from the original signer or the designated verifier performed this signature, even if he knows all secrets.

4 Modified Scheme (Probabilistic)

In this section, we modify our basic scheme from deterministic to probabilistic so as to provide the security requirement of signer's forward security.

The system setting and key generation are the same as the basic scheme.

Signature generation: When *Alice* wants to sign a message $m \in \{0,1\}^*$ while designates *Bob* to be the verifier. *Alice* executes the *Sign* algorithm and does the following steps:

- Pick $r \stackrel{R}{\leftarrow} Z_q^*$ and $Q \leftarrow rP$.
- Given r, *Alice*'s private key a, and *Bob*'s public key V_b, compute $(a+r)V_b$.
- Compute $h((a+r)V_b)$, $\mathcal{H}(m)$ and $\varsigma \leftarrow h((a+r)V_b)\mathcal{H}(m)$.
- The SDVS for m is $\sigma \leftarrow (Q, \varsigma)$.

Verification: With the knowledge that the signature σ is signed by *Alice*, then only *Bob* can verify the validity of the signature. *Bob* executes the $Veri$ algorithm and does the following steps:

- Given Q, *Bob*'s private key b and *Alice*'s public key V_a, compute $b(V_a+Q)$ and $h(b(V_a+Q))$.
- Given the message m, compute $\mathcal{H}(m)$ and $\tilde{\varsigma} \leftarrow h(b(V_a+Q))\mathcal{H}(m)$.
- Accept σ as a valid signature if and only if $\varsigma = \tilde{\varsigma}$.

4.1 Security Reduction

The modified scheme provides the same security requirement as our basic scheme. In addition, this scheme provides signer's forward security so the consistency of a signature cannot be verified by any third party even if he/she knows a signer's private key. With this property, the secrecy of previously signed signature will not be affected and a signer's privacy can be protected even if a signer's private key is disclosed. Unfortunately, although the property is important, it has been less considered in the previously proposed SDVS schemes.

The proof of non-transferability is straightforward since both entities are capable of creating such a signature so no any third party can distinguish a signer form a designated verifier, even if he knows all secrets. The security proofs of unforgeability against existential forgery under adaptive chosen message attack and indistinguishability of signer's identity can be reduced to the security of the basic scheme.

Theorem 4. Unforgeability: If there exists an adversary $\mathcal{A}$ which can break our modified SDVS scheme via EF-ACMA defined in Definition 5, then there exists another adversary $\mathcal{A}'$ which can break our basic scheme via the same attack.

This is trivial since if an adversary $\mathcal{A}$ can successfully forge a message/signature pair $(m^*, \sigma \leftarrow (Q, \varsigma))$ with a signer's public key V_i, and a designated verifier's public key V_j, then (m^*, ς) is a valid message/signature pair of our basic scheme where the signer's public key becomes $V_i + Q$ and the designated verifier's public key is V_j. Thus, we found a contradiction.

Theorem 5. Privacy of signer's identity: If there exists an adversary $\mathcal{A}$ which can break the indistinguishability of signer's identity of our modified scheme with non-negligible probability, then there exists another adversary $\mathcal{B}$ which can break the indistinguishability of signer's identity of our basic scheme with the same probability.

Proof:(sketch) We show how $\mathcal{B}$ can break our basic scheme by utilizing $\mathcal{A}$.

- $\mathcal{B}$ is given all the public information of our basic scheme constructed by an challenger $\mathcal{L}$. $\mathcal{B}$ is allowed to access the $\mathcal{H}$-oracle and signing oracle of the basic scheme which are controlled by $\mathcal{L}$. $\mathcal{B}$'s purpose is to break the indistinguishability of signer's identity of our basic scheme.
- $\mathcal{A}$ is given the same public information as $\mathcal{B}$ and is also allowed to make $\mathcal{H}$ queries and signing queries of the modified scheme, where $\mathcal{B}$ is responsible for replying these queries. $\mathcal{A}$'s purpose is to break the indistinguishability of signer's identity of the modified scheme.
- Any time when $\mathcal{A}$ makes a $\mathcal{H}$ query by providing a signer's public key pk_i, a designated verifier's public key pk_j, and a message m, $\mathcal{B}$ in turn asks to the $\mathcal{H}$-oracle by selecting $Q \stackrel{R}{\leftarrow} G$, $pk'_i \leftarrow pk_i + Q$, and providing $< m, pk'_i, pk_j >$ to the $\mathcal{H}$-oracle. The reply ς from the $\mathcal{H}$-oracle and Q is assigned as the reply of $\mathcal{A}$'s query (i.e., $\sigma \leftarrow (Q, \varsigma)$).
- At the end, when $\mathcal{A}$ requests its full-anonymity challenge in the modified scheme by providing
  - two signer's public keys pk_{sign_0}, pk_{sign_1},
  - one designated verifier's public key pk_{veri}, and
  - a message M

  to $\mathcal{B}$, then $\mathcal{B}$ in turn requests his full-anonymity challenge in the basic scheme by selecting $Q' \stackrel{R}{\leftarrow} G$, computing $pk'_{sign_0} \leftarrow pk_{sign_0} + Q'$, $pk'_{sign_1} \leftarrow pk_{sign_1} + Q'$, and providing
  - two signer's public keys $pk'_{sign_0}, pk'_{sign_1}$,
  - one designated verifier's public key pk_{veri}, and
  - a message M

  to the challenger $\mathcal{L}$. The output from $\mathcal{L}$ (which is ς') and Q' is thus the output (i.e., $\sigma^* \leftarrow (Q', \varsigma)$) from $\mathcal{B}$ to $\mathcal{A}$.
- Finally, if $\mathcal{A}$ can solve his challenge with non-negligible probability and reveals the signer's public key pk_{sign^*} of the modified scheme, then $pk'_{sign^*} = pk_{sign^*} + Q'$ is thus the signer's public key of our basic scheme. Therefore $\mathcal{B}$ also solves his challenge with non-negligible probability. □

The following theorem proves that to break the signer's forward security in this modified scheme implies to break the BDDH problem in G.

Theorem 6. Signer's forward security: If there exists an algorithm $\mathcal{A}$ which can $(t, q_{\mathcal{H}}, q_{\Sigma}, \epsilon)$-break the signer's forward security, then there exists another algorithm $\mathcal{F}$ which can $(t+q_{\mathcal{H}}O(1)_{\mathcal{H}}+q_{\Sigma}O(1)_{\Sigma}, \epsilon)$-break the $BDDH$ Problem in G. Here h-query and PrivateKey query are omitted since they are not controlled by $\mathcal{F}'$ in the following proof.

Proof: We need only to prove that any adversary without the knowledge of r or *Bob*'s private key V_b cannot verify the consistency of a *Bob*-designated signature signed by *Alice*, even if he/she knows *Alice*'s private key a. This can be proved using the same technique in the proof of Theorem 2. Here $\mathcal{A}$'s purpose is to distinguish a $Q^* \in \{Q_0, Q_1\}$ in a signature $\sigma^* \leftarrow h((a + r^*)V_b)\mathcal{H}(m)$, $r^* \in \{r_0, r_1\}$, by providing $< a, V_b, m, Q_0(= r_0P), Q_1(= r_1P) >$ to $\mathcal{F}$ at its choice. $\mathcal{F}$'s purpose is to find the solution of a BDDH problem by providing two three-tuple $< V_a + Q_0, V_b, \mathcal{H}(m) >$ and $< V_a + Q_1, V_b, \mathcal{H}(m) >$ to the outsider $\mathcal{L}$. Finally, the challenge outputted from $\mathcal{L}$ is assigned to $\mathcal{A}$ by $\mathcal{F}$ as $\mathcal{A}$'s challenge σ^*. Therefore, $\mathcal{F}$ solves its BDDH challenge successfully if $\mathcal{A}$ outputs a correct $Q^* \in \{Q_0, Q_1\}$. We omit this proof (see details of the proof of Theorem 2).

5 Efficiency and Performance Comparison

In this section, we give the comparison results of our schemes with other schemes in efficiency and performance, which are shown in Table 1. In Table 1, in order for the results to be compared effectively, we only consider the operations of **P**airing computation (P), **E**lliptic **C**urve **M**ultiplication (ECM), **Exp**onentional computation (Exp) and **Inv**ersive operation (Inv), which are the most time-consuming operations. In addition, the size of n^r is about 111 bits ($|n^r| \approx 111$) according to [7] and we can set $|p| = 512$, $|G| = |Z_q|$ with $|q| = 160$ in Table 1 so that they can have comparable security.

In out basic scheme, since the randomize is depended on the hash $\mathcal{H}(m)$ of a message m instead of a random parameter, it realized the low communication cost. The data flow of our basic scheme consists of only one parameter in G, while previously proposed (strong)DVS schemes consists of at least two parameters. On the other hand, this scheme is very efficient in computation. If we neglect the hashing operations which does not cost a lot of time, then the time-consuming operations in this scheme consists of only two multiplicative operations on G for each signer and verifier. In addition, one of the two operations can be pre-computed off-line. Our modified scheme is as efficient as the basic scheme except that all of the computation has to be computed on-line. The data flow increases to *two* parameters in G but this sacrifice increased its security.

From the performance comparison, we conclude that our schemes are superior to other schemes in almost every aspects.

6 Designated Verifier Signcryption

In some circumstances, in addition to preserve the privacy of a sender *Alice*, she may also wish (or asked) to encrypt the message so as to avoid any third party to read it (if the message is concerned with confidential matters). Obviously, this can be done by encrypting the message using a designated verifier's public key or combining a DVS scheme with any key agreement scheme. This, however, requires additional complex operations and sometimes increases the data flow in number. In our scheme, it is obvious that if a message m is a secret, than

Table 1. Performance Comparison I

	Data Flow	Sign		Verify		Type	Forward Security
		off-line	on-line	off-line	on-line		
Basic Scheme	1 **in** G	1 **ECM**	1 **ECM**	1 **ECM**	1 **ECM**	**SDVS**	No
Mod. Scheme	2 **in** G	–	2 **ECM**	–	2 **ECM**	**SDVS**	**Yes**
JIS [6]	3 in Z_q 3 in Z_p	–	4 ECM	–	4 ECM	DVS	No
LV [7]	1 in n_r 1 in Z_q	–	1 P	–	1 P 1 ECM	SDVS	No
SKM [11]	3 in Z_q	–	1 Exp 1 Inv	–	3 Exp	SDVS	No
SZM [13]	2 in Z_q 1 in G	–	1 P 3 ECM	1 P	2 P 2 Exp	SDVS	No

$h(m)$ is also a secret from the viewpoint of a third party. But *Alice* and *Bob* can get $h(m)$ because they both know the value $aV_b = abP = bV_a$. Consequently, $h(m)$ can be used as a session key in our scheme to encrypt m with virtually no additional cost.

To avoid a deterministic encryption of any message m, we concatenate m with a random number $r \in \{0,1\}^l$ where l is a security parameter for a symmetric encryption algorithm E. In the following algorithm, we introduce our designated verifier signcryption algorithm using our basic scheme. We emphasize that our modified scheme can also be changed into a signcryption scheme using the same technique.

Assume the system setting and key generation are the same as those in Section 3, then, when *Alice* wants to signcrypt a message $m \in \{0,1\}^*$ to B, he does as follows:

Signcryption:

- Given *Alice*'s private key a, and *Bob*'s public key V_b, compute aV_b.
- Pick $r \overset{R}{\leftarrow} \{0,1\}^l$, compute $h(aV_b)$ and $\mathcal{H}(m||r)$.
- Compute $\sigma \leftarrow h(aV_b)\mathcal{H}(m||r)$.
- Compute $k \leftarrow \mathcal{H}(m||r)$.
- Compute $c \leftarrow E_k(m||r)$.

Then the signcryption for message m is (c, σ).

Un-signcryption: Knowing that the signcrypted message is originated from *Alice*, then only *Bob* can decrypt c and verify the validity of the signature σ.

- Given *Bob*'s private key b, and *Alice*'s public key V_a, compute bV_a.
- Compute $h(bV_a)^{-1}$.
- Compute $\widetilde{k} \leftarrow h(bV_a)^{-1}\sigma$
- Compute $\widetilde{m||r} \leftarrow D_{\widetilde{k}}(c)$

Table 2. Performance Comparison II

	Data Flow	Signcrypt		Unsigncrypt		Forward
		off-line	on-line	off-line	on-line	Security
Basic Scheme	2	1 **ECM**	1 **ECM**	1 **ECM** 1 **Inv**	1 **ECM**	No
Mod. Scheme	3	–	2 **ECM**	1 **Inv**	2 **ECM**	**Yes**
SKM [11]	4	–	1 Exp 1 Inv	–	3 Exp	No
Zheng [14]	3	–	1 Exp 1 Inv	–	2 Exp	No

- Extract $\tilde{m}$ from $\widetilde{m||r}$

where D is the corresponding decryption algorithm of E and Bob accepts the message $\tilde{m} = m$ and the signature if and only if $\mathcal{H}(\widetilde{m||r}) = \widetilde{k}$.

The security of this signcryption scheme is depended on the security of the proposed SDVS scheme and the symmetric encryption scheme E. Also note that without knowing $h(aV_b)$ or m, the probability of extracting a correct $k = h(m||r)$ from σ is $1/q$, which is negligible.

Table 2 shows the performance comparison of our scheme with two previously proposed schemes. The scheme proposed in [11] is also a designated verifier signcryption scheme modified from a SDVS scheme.

7 conclusion

In this paper, we first introduced a simple SDVS scheme which is much more efficient than previously proposed SDVS schemes. Based on this scheme, we proposed our modified scheme which provides an additional security requirement called signer's forward security. With this additional property, the secrecy of previously signed signature will not be affected and a signer's privacy can be protected even if a signer's private key is disclosed. In addition, our schemes can be modified into designated verifier signcryption schemes with virtually no additional cost comparing to the original SDVS schemes, which is useful in the case if the message is required to be kept secret from any third party. In the appendix, we give concrete security proofs to show that our schemes are provable secure in the random oracle model.

References

1. Y. Aumann, and M. Rabin, *Efficient deniable authentication fo long messages*, International Conference on Theoretical Computer Science in Honor of Professor Manuel Blum's 60th Birthday (1998), avaible at **http://www.cs.cityu.edu.hk/dept/video.html**.

2. D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, Advances in cryptology –CRYPTO'01, Lecture Notes in Comput Sci. **2139** (2001), 213–229.
3. D. Boneh, B. Lynn and H. Shacham, *Short signatures from the Weil pairing*, Advances in cryptology –CRYPTO'01, Lecture Notes in Comput Sci. **2248** (2001), 514–532.
4. L. Fan, C. X. Xu, and J. H. Li, *Deniable authentication protocol based on Diffie-Hellman algorithm*, Electronics Letters **38(4)** (2002), 705–706.
5. S. Goldwasser, S. Micali and R. L. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of Computing **17(2)** (1988), 281–308.
6. M. Jakkobsson, K. Sako and T. Impagliazzo, *Designated verifier proofs and their applications*, Advances in cryptology –EUROCRYPTO'96, Lecture Notes in Comput Sci. **1070** (1996), 143–154.
7. F. Laguillaumie, D. Vergnaud, *Designated verifier signatures: anonymity and efficient construction from any bilinear map*, SCN'04, Lecture Notes in Comput Sci. **3352** (2005), 107–121.
8. L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, *An efficient protocol for authenticated key agreement*, Designs, Codes and Cryptogr. **28** (2003), no. 2, 119–134.
9. T. Okamoto, R. Tso and E. Okamoto, *One-way and two-party authenticated ID-based key agreement protocols using pairing*, MDAI'05, Lecture Notes in Artificial Intelligence **3558** (2005), 122–133.
10. R. L. Rivest, A. Shamir and Y. Tauman, *How to leas a secret*
11. S. Saeednia, S. Kremer and O. Markowitch, *An efficient strong designated verifier signature scheme*, ICISC'03, Lecture Notes in Comput Sci. **2971** (2003), 40–54.
12. Z. Shao, *Efficient deniable authentication protocol based on generalized ElGamal signature scheme*, Computer Standards & Interfaces, **26** (2004), 449–454.
13. W. Susilo, F. Zhang and Y. Mu, *Identity-based strong designated verifier signature schemes*, ACISP'04, Lecture Notes in Comput Sci. **3108** (2004), 313–324.
14. Y. Zheng, *Digital signcryption or how to achieve cost(signature* & *encryption)* $<<$ *cost(signature)+cost(encryption)*, Advances in cryptology –CRYPTO'97, Lecture Notes in Comput Sci. **1294** (1997), 165–179.