Short Designated Verifier Proxy Signature from Pairings*

Xinyi Huang[1], Yi Mu[2], Willy Susilo[2], and Futai Zhang[1,**]

[1] College of Mathematics and Computer Science,
Nanjing Normal University, P.R. China
xinyinjnu@126.com, zhangfutai@njnu.edu.cn
[2] Centre for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong, Australia
{wsusilo, ymu}@uow.edu.au

Abstract. In a designated verifier proxy signature scheme, the original signer delegates her/his signing capability to the proxy signer in such a way that the latter can sign messages on behalf of the former, but only the designated verifier can believe the validity of these signatures. In this paper, we firstly describe the notion of short designated verifier proxy signature, which we call SDVPS. Then a concrete scheme is presented. We prove that the proposed scheme is unforgeable even to the original signer under the Gap Bilinear Diffie-Hellman assumption and Random Oracle Model.

Keywords: Proxy Signature, Short Signature, Pairings, Authentication.

1 Introduction

In a proxy signature scheme, the original signer (say, Alice) can delegate her signing right to another user (say, Bob) who is called proxy signer. Bob can sign messages on behalf of Alice. Upon receiving a proxy signature on some message, the verifier can validate its correctness by a given verification procedure and can be convinced of the original signer's agreement on the proxy signing. The notion of proxy signature was introduced in [7]. Proxy signature schemes have been suggested for use in a number of applications, including electronic commerce and distributed shared object systems. Based on the application, they can be classified as *full delegation*, *partial delegation*, and *delegation by warrant* schemes. Based on the knowledge of the proxy private key, proxy signatures can be classified into *proxy-unprotected* and *proxy-protected*. In a proxy-protected scheme only the proxy signer can generate proxy signatures, while in a proxy-unprotected scheme either the proxy signer or the original signer can generate

* This work is supported by ARC Discovery Grant DP0557493.
** Partially supported by Ministry of Education of Jiangsu Province Project 03KJA520066 and Open Project of Key Laboratory on Computer Network and Information Security of Ministry of Education of China.

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 835–844, 2005.

proxy signatures since both of them have a knowledge on the proxy private key. In many applications, proxy-protected schemes are required to avoid the potential disputes between the original signer and the proxy signer.

There have been several interesting works that provide different features to proxy signature, for example, threshold proxy signature [15], one-time proxy signature [13], ID-based proxy signature [14], etc. Let's consider a scenario where the proxy signer wishes to protect his signing privilege from knowing by other parties. That is, Bob only wants to convince the designated receiver that he has signed the specific message. This scenario is related to the *designated verifier* signatures proposed by Jakobsson, Sako and Impagliazzo in [4]. This signature scheme can be considered as the first non-interactive undeniable signature scheme that transforms Chaum's scheme [1] into non-interactive verification using a designated verifier proof. In a designated verifier scheme, the signature provides authentication of a message without providing a non-repudiation property of traditional signatures. A designated verifier scheme can be used to convince a single third party, i.e., the designated verifier, and only the designated verifier can be convinced about its validity or invalidity. This is due to the fact that the designated verifier can always create a signature intended for himself that is indistinguishable from an original signature. This scheme does not require any interaction with the presumed signer to verify the authenticity of the message.There are a number of other works on designated verifier signatures, for example [5, 4, 9, 10, 8, 11].

Constructing an ordinary designated verifier proxy signature scheme is trivial (e.g., [2],[12]). The motivation of this paper is to find a scheme of designated verifier proxy signature which is *very short*. We call it Short Designated Verifier Proxy Signature (SDVPS). Compared with other schemes, our proxy key generation is *noninteractive* and the signature length is *shortest*. We prove that our scheme is proxy-protected that is even the original signer cannot forge a valid signature. The proof is based on the Gap Bilinear Diffle-Hellman problem in random oracle.

The rest of this paper is organized as follows. In the next section, we will provide some preliminaries and background required throughout the paper. In Section 3, we introduce the notion of the SDVPS scheme. In Section 4, we provide our concrete SDVPS scheme, and its security proof is given in Section 5. In Section 6, we compare the performance of our scheme with the existing scheme. Section 7 concludes this paper.

2 Preliminaries

In this section, we will review some fundamental backgrounds required in this paper, namely bilinear pairing and the definition of the designated verifier signature.

2.1 Basic Concepts on Bilinear Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic additive groups generated by P_1, P_2, respectively, whose orders are a prime q. Let $\mathbb{G}_M$ be a cyclic multiplicative group with the same

order q. We assume there is an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(P_2) = P_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_M$ be a bilinear mapping with the following properties:

1. *Bilinearity*: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b, \in \mathbb{Z}_q$.
2. *Non-degeneracy*: There exists $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1_{\mathbb{G}_M}$.
3. *Computability*: There exists an efficient algorithm to compute $e(P, Q)$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$.

For simplicity, hereafter, we set $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$. We note that our scheme can be easily modified for a general case, when $\mathbb{G}_1 \neq \mathbb{G}_2$.

2.2 Complexity Assumptions

We assume that the Bilinear Diffie-Hellman problem is intractable in polynomial time. Formally, we define it as follows.

Definition 1. Bilinear Diffie-Hellman (BDH) Problem
Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$), compute $e(P, P)^{abc}$.

Definition 2. Decisional Bilinear Diffie-Hellman (DBDH) Problem
Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP, cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$) and $h \in \mathbb{G}_M$, decide whether $h = e(P, P)^{abc}$.

Definition 3. Gap Bilinear Diffie-Hellman (GBDH) Problem
Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$), compute $e(P, P)^{abc}$ with the help of the DBDH oracle.

2.3 Designated Verifier Signature

The goal of designated verifier proofs is to allow an entity, Alice, to prove the validity of a statement Θ to a specific entity, Bob, in such a way that Bob is convinced about this fact but he cannot transfer this conviction to other third party. In [4], it is suggested that Alice should prove the statement "Θ is correct or I know Bob's secret key". Bob, who is aware that he has not generated the proof himself and also sure that Alice does not know his secret key will be convinced by this proof (i.e. the first part of the proof, namely Θ is correct), while no other verifier can decide which part of the disjunction is correct.

The notion of designated verifier proofs are given in [4], and they are formalized in [8] as follows.

Definition 1. Designated Verifier Signature [8]
Let $P(A, B)$ be a protocol between Alice and Bob so that Alice can prove the correctness of statement Θ. Bob is said to be a designated verifier if he can produce identically distributed transcripts that are indistinguishable from those of $P(A, B)$.

3 Short Designated Verifier Proxy Signature(SDVPS)

3.1 Outline of the SDVPS

There exist three participants in the system, namely Alice, Bob and Cindy, who act as the original signer, the proxy signer and the receiver (or the designated verifier), respectively. We denote (x_i, P_i) as a pair of private key and public key for user i, where $i \in \{A, B, C\}$ indicating Alice, Bob, and Cindy, respectively. A short designated verifier proxy signature scheme (SDVPS) consists of following six essential algorithms:

- ParamGen: It takes as input the system security parameter ℓ and outputs the system parameters.
- KeyGen: It takes as input the security parameter ℓ and outputs the key set: (x_i, P_i) for $i = A, B, C$.
- ProxyKeyGen: A deterministic algorithm that takes as input the original signer's secret key, the proxy signer's secret key, the identity of the proxy signer and the warrant m_w to generate the proxykey. That is proxykey $\leftarrow$ ProxyKeyGen(x_A, x_B, ID_B, m_w). where x_A, x_B is the secret key of the original signer and the proxy signer, ID_B is the identity of the proxy signer.
- Sign: A deterministic algorithm that takes as input the proxykey, the designated verifier's public key and a message m to generate a signature σ. That is $\sigma \leftarrow$ Sign(proxykey, ID_B, P_C, m), where proxykey is generated by the above ProxyKeyGen algorithm, ID_B is the identity of the proxy signer and P_C is the public key of the receiver(the designated verifier).
- Verify: A deterministic algorithm that accepts a message m, a signature σ, the original signer's public key P_A, the proxy signer's public key P_B, the proxy signer's identity and the receiver's secret key x_c and returns True if the signature is correct, or $\perp$ otherwise. That is, $\{\text{True}, \perp\} \leftarrow$ Verify$(P_A, P_B, ID_B, x_C, m, \sigma)$.
- Transcript Simulation: An algorithm that is run by the verifier to produce identically distributed transcripts that are *indistinguishable* from the original protocol.

In addition to the above main algorithms, we also require the following.

- **Correctness.** All signatures generated correctly by Sign algorithm must always pass the verification algorithm. That is,

$$Pr\left(True \leftarrow \text{Verify}(P_A, P_B, ID_B, x_C, m, \text{Sign}(\text{proxykey}, ID_B, P_C, m), m_w)\right) = 1.$$

- **Transcript Simulation Generation.** We require that the verifier, who holds the secret key x_C can always produce identically distributed transcripts that are indistinguishable from the original protocol via the Transcript Simulation algorithm.

3.2 Security Model

There are three types adversaries in the system:

1. **Type I:** This type adversary only has the public keys of Alice and Bob.
2. **Type II:** This type of adversary has the public keys of Alice and Bob, her/he also has the secret key of Bob (the proxy signer).
3. **Type III:** This type of adversary has the public keys of Alice and Bob, her/he also has the secret key of Alice (the original signer).

We can find that if our short proxy signature scheme is unforgeable against Type II (or Type III) adversary, our scheme is also unforgeable against Type I adversary.

Formal Security Notion: Unforgeability of the SDVPS

We provide a formal definition of existential unforgeability of a short designated verifier proxy signature scheme (SDVPS) under a chosen message attack(EF-CMA-adversary). It is defined using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Setup: $\mathcal{C}$ runs the algorithm to generate the public keys (P_A, P_B and P_C) of the original signer A, proxy signer B and the designated verifier C. $\mathcal{C}$ also generates the identity ID_B of the proxy signer.
- Sign Queries: $\mathcal{A}$ can request a proxy signature on a message m with the original signer A, the proxy signer B and the designated verifier C. In response, $\mathcal{C}$ outputs a signature σ for a message m.
- Verify Queries: $\mathcal{A}$ can request a signature verification on a pair (m, σ) with the original signer A, the proxy signer B and the designated verifier C. In response, $\mathcal{C}$ outputs True if it is correct, or $\perp$ otherwise.
- Output: Finally, $\mathcal{A}$ outputs a new pair (m^*, σ^*), where m^* has never been queried during the Sign Queries and σ^* is a valid signature for the original signer A, the proxy signer B and the designated verifier C.

The success probability of an adversary to win the game is defined by

$$Succ^{EF-CMA}_{SDVPS,\mathcal{A}}(\ell).$$

Definition 4. *We say that a short designated verifier proxy signature scheme is existentially unforgeable under a chosen message attack if the probability of success of any polynomially bounded adversary in the above game is negligible for all the three types of adversaries. In other words,* $Succ^{EF-CMA}_{SDVPS,\mathcal{A}}(\ell) \leq \epsilon$ *where* $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}, \mathcal{A}_{III}\}$ *and* ϵ *is negligible.*

4 Our SDVPS Scheme

As assumed earlier, there are three participants in the system, namely Alice, Bob and Cindy, who act as the original signer, the proxy signer and the receiver (or the designated verifier), respectively. Our SDVPS consists of the following algorithms.

1. ParamGen: Taking as input the system security parameter ℓ, the algorithm outputs $\{\mathbb{G}_1, \mathbb{G}_M, q, e, P\}$, including a cyclic additive group $\mathbb{G}_1$ of order $q(q \geq 2^\ell)$, a multiplicative group $\mathbb{G}_M$ of order q, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_M$ and a generator P of $\mathbb{G}_1$. This algorithm also outputs two cryptographic hash functions H_0 and H_1 where $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$.
2. KeyGen: Taking as input the system security parameter k, the algorithm outputs three pairs of secret/public keys $(x_i, P_i = x_i P)$, for $i = A, B, C$, which denote Alice, Bob, and Cindy, respectively.
3. ProxyKeyGen:
   (a) Alice computes $D_{AB} = x_A Q_B$, where $Q_B = H_0(ID_B, P_B, m_w)$, ID_B is the identity of Bob, P_B is the public key of Bob, and m_w is the warrant. Alice then sends (D_{AB}, m_w) to Bob.
   (b) Bob verifies whether $e(D_{AB}, P) = e(Q_B, P_A)$ holds.
   (c) Bob obtains the proxykey (x_B, D_{AB}).
4. Sign: For a message m, Bob computes $\sigma = H_1(m, e(D_{AB} + x_B Q_B, P_C))$ and the designated verifier proxy signature on the message m is σ.
5. Verify: To check whether σ is a valid signature of the message m and the warrant m_w, Cindy uses her secret key x_C to check: $\sigma \stackrel{?}{=} H_1(m, e(x_C Q_B, P_A + P_B))$ where $Q_B = H_0(ID_B, P_B, m_w)$. If the above equation holds, Cindy accepts the signature σ, otherwise rejects it.
   Correctness:

$$H_1(m, e(x_C Q_B, P_A + P_B) = H_1(m, e(x_C Q_B, x_A P + x_B P))$$
$$= H_1(m, e((x_A + x_B) Q_B, x_C P)) = H_1(m, e(D_{AB} + x_B Q_B, P_C))$$

   Transcript Simulation:
   Cindy can use her secret key to compute an arbitrary signature on a message m^* as $\sigma^* = H_1(m^*, e(x_C Q_B, P_A + P_B))$.

5 Security Analysis

In this section, we will firstly prove that the proposed scheme is a designated verifier signature scheme. Then we prove that our SDVPS is secure against all types of adversaries.

Theorem 1. *The proposed scheme is a designated verifier signature scheme.*

Proof: For any message m, Cindy can compute a valid signature by computing $\sigma = H_1(m, e(x_C Q_B, P_A + P_B))$. One can find that signature generated like this is the same as the original one generated by the proxy signer Bob. Therefore, even given Cindy's secret key x_C, no one can believe the signature is sent by Bob.

Theorem 2. *If the Type II Adversary $\mathcal{A}_{II}$ (the proxy signer Bob) can forge a valid signature of the proposed scheme with success probability $Succ^{EF-CMA}_{SDVPS, \mathcal{A}_{II}}$ after making q_H queries to the $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ ($q \geq 2^\ell$, where ℓ is the system's security parameter), q_S queries to the signing algorithm and q_V to the verifying*

algorithm in polynomial time t, then there exists an algorithm $\mathcal{B}$ who can use $\mathcal{A}_{II}$ to solve an instance of the GBDH problem with probability: $Succ_{\mathcal{B}}^{GBDH} \geq Succ_{SDVPS,\mathcal{A}_{II}}^{EF-CMA} - \frac{q_V}{2^{\ell}-q_H-q_S}$ in the same time t.

Proof: Our overall strategy for the proof is as follows. We shall define a sequence Game_0, Game_1, Game_2, Game_3, Game_4 of attack games. Each game operates on the same underlying probability space, in particular, the system's parameter, public keys of the original signer Alice, the proxy signer Bob, the receiver Cindy and the values of the random oracle $\mathcal{H}$. We will prove that if there exists $\mathcal{A}_{II}$ who can forge a valid signature of our SDVPS scheme, then there exists $\mathcal{B}$ who can use $\mathcal{A}_{II}$ to solve an instance of Gap Bilinear Diffle-Hellman problem. That is given a random instance (P, aP, bP, cP), $\mathcal{B}$ can use $\mathcal{A}_{II}$ to obtain the value of $e(P,P)^{abc}$ with the help of Decisional Bilinear Diffle-Hellman(DBDH) Oracle.

$\mathcal{B}$ will simulate all the oracles in the proof. In the simulation, $\mathcal{B}$ will maintain a list which is called *H-List* to record the hash queries and the corresponding values. We assume that $\mathcal{A}_{II}$ is well-behaved in the sense that $\mathcal{A}_{II}$ will never repeat the same queries in the simulation.

- Game_0. We consider a Type II *EF-CMA* adversary $\mathcal{A}_{II}$ with the success probability $Succ_{SDVPS,\,\mathcal{A}_{II}}^{EF-CMA}$. The original signer, Alice, selects his secret key $x_A \in \mathbb{Z}_q^*$ and sets his public key as $P_A = x_A P$. The proxy signer Bob and designated verifier Cindy also generate their own secret/public key pairs (x_B, P_B) and (x_C, P_C). Bob also publishes his identity ID_B.
  The adversary $\mathcal{A}_{II}$, fed with (P_A, P_B, P_C) and x_B, can query the hash oracle H, the signing algorithm and the verify algorithm, and outputs (m^*, σ^*), such that$\mathsf{Verify}(P_A, P_B, ID_B, x_C, m^*, \sigma^*) = \mathsf{True}$.
  Let q_H, q_S, q_V denote the numbers of queries to the H, signing algorithm and verifying algorithm. The requirement is that m^* cannot be queried to the signing algorithm.
  In any Game_i, we denote by Forge_i the event $\mathsf{Verify}(P_A, P_B, ID_B, x_C, m, \sigma) = \mathsf{True}$. By definition, we have $\Pr[\mathsf{Forge}_0] = Succ_{SDVPS,\,\mathcal{A}_{II}}^{EF-CMA}$.
- Game_1. In this game, $\mathcal{B}$ sets $P_A = aP$, $Q_B = bP$ and $P_C = cP$ where aP, bP, cP are the random instance of the Gap Bilinear Diffle-Hellman problem. $\mathcal{B}$ also chooses $b' \in \mathbb{Z}_q^*$ and sets $P_B = b'P$. Then $\mathcal{B}$ returns (P_A, P_B, P_C, Q_B, b') to $\mathcal{A}_{II}$. Since a, b, c, b' are randomly chosen, therefore $Pr[\mathsf{Forge}_1] = Pr[\mathsf{Forge}_0]$
- Game_2. In this game, B will simulate the random oracle H. There is a list *H-List* which maintains all the queries and answers consists of tuple $(m_i, r_i, \sigma_i, coin_i)$. Here (m_i, r_i) is the input of the H and σ_i is the output of H. $coin_i = 1$ if $r_i \cdot e(-P_C, Q_B)^{b'} = e(P,P)^{abc}$ and $coin_i = 0$ otherwise. For any query (m_i, r_i) to the oracle H, $\mathcal{B}$ submits $(r_i \cdot e(-P_C, Q_B)^{b'}, aP, bP, cP)$ to the DBDH oracle and DBDH oracle will tell $\mathcal{B}$ whether $r_i \cdot e(-P_C, Q_B)^{b'} = e(P,P)^{abc}$ or not
  1. If $r_i \cdot e(-P_C, Q_B)^{b'} = e(P,P)^{abc}$, $\mathcal{B}$ sets $coin_i = 1$ and checks the *H-List*
     (a) If there exists an item $(m_i, \perp, \sigma_i, 1)$ in the *H-List*, $\mathcal{B}$ returns σ_i as the answer.

(b) Otherwise, $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot,\cdot,\sigma_i,\cdot)$ in the *H-List*. $\mathcal{B}$ then adds $(m_i, r_i, \sigma_i, 1)$ into the *H-List* and returns σ_i as the answer.

2. If $r_i \cdot e(-P_C, Q_B)^{b'} \neq e(P,P)^{abc}$, $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot,\cdot,\sigma_i,\cdot)$ in the *H-List*. $\mathcal{B}$ then adds $(m_i, r_i, \sigma_i, 0)$ into the *H-List* and returns σ_i as the answer.

In the random oracle model, this game is clearly identical to the previous one. Hence $Pr[\mathsf{Forge}_2] = Pr[\mathsf{Forge}_1]$.

– Game_3. In this game, $\mathcal{B}$ simulates the signing algorithm. After receiving $\mathcal{A}_{II}$'s choice of the message m_i, B performs:

1. If there is a triple $(m_i, r_i, \sigma_i, 1)$ in the *H-List*, $\mathcal{B}$ outputs σ_i as the signature.
2. Else $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot,\cdot,\sigma_i,\cdot)$ in the *H-List*. Then $\mathcal{B}$ adds $(m_i, \perp, \sigma_i, 1)$ to the *H-List* and outputs σ_i as the answer.

Then $\mathcal{A}_{II}$ gets the value σ_i as the signature of m_i. Of course, this oracle simulates the signature perfectly, so $\Pr[\mathsf{Forge}_3] = \Pr[\mathsf{Forge}_2]$.

– Game_4. In this game, $\mathcal{B}$ simulates the verifying algorithm. After receiving $\mathcal{A}_{II}$'s request of (m_i, σ_i), $\mathcal{B}$ checks :

1. If there is no item $(\cdot,\cdot,\sigma_i,\cdot)$ in the *H-List*, $\mathcal{B}$ rejects (m_i, σ_i) as an invalid signature.
2. Else, there is an item $(\cdot,\cdot,\sigma_i,\cdot)$ in the $H - List$:
   (a) If this item has the form of $(m_i, \perp, \sigma_i, 1)$ or $(m_i, r_i, \sigma_i, 1)$, $\mathcal{B}$ will accept it as a valid signature.
   (b) Otherwise, $\mathcal{B}$ rejects it as an invalid signature.

This makes a difference only if (m_i, σ_i) is a valid signature, while σ_i is not queried from the oracle H. Since, H is uniformly distributed, this case happens with probability less than $\frac{1}{2^\ell - q_H - q_S}$. Summing up for all verifying queries, we get $\Pr[\mathsf{Forge}_3] - \Pr[\mathsf{Forge}_4] \leq \frac{q_V}{2^\ell - q_H - q_S}$.

After Game_4 terminates, $\mathcal{A}_{II}$ outputs a valid signature (m^*, σ^*) such that

$$\mathsf{Verify}(P_A, P_B, ID_B, x_C, m^*, \sigma^*) = \mathsf{True}.$$

That is, there is an item $(\cdot,\cdot,\sigma^*,\cdot)$ in the *H-List*. By the definition of the EF-CMA adversary model, m^* can not be queried in the sign oracle, so σ^* is returned as the hash value of $\mathcal{A}'_{II}s$ query (m^*, r^*). That is to say $(m^*, r^*, \sigma^*, 1)$ is in the *H-List* and $r^* \cdot e(P_C, -Q_B)^{b'} = e(P,P)^{abc}$. Note that $P_C = cP, Q_B = bP$ and b' is randomly chosen by $\mathcal{B}$, so $\mathcal{B}$ can compute $e(P,P)^{abc} = r^* \cdot e(bP, -cP)^{b'}$. Therefore, given aP, bP, cP, $\mathcal{B}$ successfully solves an instance of the GBDH problem with probability: $Succ_{\mathcal{B}}^{GBDH} \geq Succ_{SDVPS,\mathcal{A}_{II}}^{EF-CMA} - \frac{q_V}{2^\ell - q_H - q_S}$. ∎

Theorem 3. *If the Type III Adversary $\mathcal{A}_{III}$ (that is the original signer Alice) can forge a valid signature of the proposed scheme with success probability $Succ_{SDVPS,\mathcal{A}_{III}}^{EF-CMA}$ after making q_H queries to the $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$($q \geq 2^\ell$, ℓ is the system's security parameter), q_S queries to the signing algorithm and q_V to the verifying algorithm in some polynomial time t , then there exists an algorithm $\mathcal{B}$ who can use $\mathcal{A}_{III}$ to solve an instance of the GBDH problem with probability: $Succ_{\mathcal{B}}^{GBDH} \geq Succ_{SDVPS,\mathcal{A}_{III}}^{EF-CMA} - \frac{q_V}{2^\ell - q_H - q_S}$ in the same time t.*

Proof. The whole proof is almost the same as the above, except that Given $aP.bP, cP$, $\mathcal{B}$ sends $(P_A = a'P, P_B = aP, Q_B = bP, P_C = cP, a')$ to this Type III adversary.

At last, $\mathcal{A}_{III}$ outputs a valid signature (m^*, σ^*) such that $\mathsf{Verify}(m^*, \sigma^*, P_A, P_B, Q_B, c) = \mathsf{True}$. That is to say $(m^*, r^*, \sigma^*, 1)$ is also in the $H - Liast$. Since σ^* is a valid signature of the message m^*, then $r^* \cdot e(P_C, -Q_B)^{a'} = e(P, P)^{abc}$. Note that $P_C = cP, Q_B = bP$ and a' is randomly chosen by $\mathcal{B}$, so $\mathcal{B}$ can compute $e(P, P)^{abc} = r^* \cdot e(bP, -cP)^{a'}$. Therefore, given aP, bP, cP, $\mathcal{B}$ successfully solves an instance of the GBDH problem with probability: $Succ_{\mathcal{B}}^{GBDH} \geq Succ_{SDVPS,\mathcal{A}_{III}}^{EF-CMA} - \frac{q_V}{2^{\ell} - q_H - q_S}$. ∎

6 Comparison

In this section, we compare the signature length of our short designated verifier signature scheme (*SDVPS*) with Wang's scheme in [12]. The signature of Wang's scheme is (r_p, K, D, s) where $r_p, K, D \in \mathbb{Z}_p$ and $s \in \mathbb{Z}_q$. Let $|\mathbb{Z}_p|$ denote the bit length of the element in $\mathbb{Z}_p$ and $|\mathbb{Z}_q|$ denote the the bit length of the element in $\mathbb{Z}_q$, we have the following table.

Scheme	**Signature Length**	$p : 1024; q : 160$
Wang's Scheme	$3\|\mathbb{Z}_p\| + \|\mathbb{Z}_q\|$	3232 bits
Our Scheme	$\|\mathbb{Z}_q\|$	160 bits

One can find that the signature length of our *SDVPS* scheme is *dramatically* decreased, which is more applicable in the networks with limited bandwidth. One can also find that the implementation of out scheme needs the *bilinear pairing*, how to get a *SDVPS* scheme without the need of pairing is an open problem.

7 Conclusion

We have presented a new designated verifier proxy signature scheme, which we believe is the shortest among all the known designated verifier proxy signatures. We prove that our scheme offers transcript simulation as a normal designated signature. We also prove that our scheme is secure under random oracle model.

References

1. D. Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology, Proc. EUROCRYPT 1991,* LNCS 547, pages 458–464. Springer–Verlag, Berlin, 1991.
2. J. Z. Dai, X. H. Yang, and J. X. Dong. Designated-receiver proxy signature scheme for electronic commerce. In *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, pages 384–389. IEEE Press, 2003.

3. S. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. In *Proc. of CT-RSA 2003,* LNCS 2612, pages 80–97. Springer–Verlag, Berlin, 2003.
4. M. Jakobsson, K.Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology, Proc. EUROCRYPT 1996,* LNCS 1070, pages 143–154. Springer–Verlag, Berlin, 1996.
5. F. Laguillaumie and D. Vergnaud. Designated verifiers signature: Anonymity and efficient construction from *any* bilinear map. In *Fourth Conference on Security in Communication Networks '04 (SCN 2004),* LNCS 3352, pages 107–121. Springer–Verlag, Berlin, 2004.
6. B. Libert and J.-J. Quisquater. Identity based undeniable signatures. In *Proc. of CT-RSA 2004,* LNCS 2964, pages 112–125. Springer–Verlag, Berlin, 2004.
7. M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *Proc. of the Third ACM Conf. on Computer and Communications Security*, pages 48–57, 1996.
8. S. Saeednia, S. Kramer, and O. Markovitch. An efficient strong designated verifier signature scheme. In *The 6th International Conference on Information Security and Cryptology (ICISC 2003),* LNCS 2971, pages 40–54. Springer–Verlag, Berlin, 2003.
9. R. Steinfeld, H. W. L. Bull, and J. Pieprzyk. Universal designated-verifier signatures. In *Advances in Cryptology–ASIACRYPT 2003,* LNCS 2893, pages 523–543. Springer–Verlag, Berlin, 2003.
10. R. Steinfeld, H. W. L. Bull, and J. Pieprzyk. Efficient extension of standard schnorr/rsa signatures into universal designated-verifier signatures. In *Public Key Cryptography, Proc. PKC 2004,* LNCS 2947, pages 86–100. Springer–Verlag, Berlin, 2004.
11. W. Susilo, F. Zhang, and Y. Mu. Identity-based strong designated verifier signature schemes. In *Proceedings of the Information Security and Privacy, 9th Australasian Conference (ACISP 2004),* LNCS 3108, pages 313–324. Springer–Verlag, Berlin, 2004.
12. G. Wang. Designated-verifier proxy signatures for e-commerce. In *the IEEE 2004 International Conference on Multimedia and Expo (ICME 2004)*, pages 1731–1734. IEEE Press, 2004.
13. H. Wang and J. Pieprzyk. Efficient one-time proxy signature. In *Advances in Cryptology–Aisacrypt 2003,* LNCS 2894, pages 507–522. Springer–Verlag, Berlin, 2003.
14. F. Zhang and K. Kim. Id-based blind signature and proxy signature from bilinear pairings. In *In: Information Security and Privacy (ACISP 2003),* LNCS 2727, pages 312–323. Springer–Verlag, Berlin, 2003.
15. K. Zhang. Threshold proxy signature schemes. In *In Proc. Information Security (ISW 1997),* LNCS 1396, pages 282–290. Springer–Verlag, Berlin, 1997.