20-7-2005

Universal designated multi verifier signature schemes

Ching Yu Ng
University of Wollongong, cyn27@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Y. Mu
University of Wollongong, ymu@uow.edu.au

Universal designated multi verifier signature schemes

Abstract

The notion of Universal Designated-Verifier Signatures was put forth by Steinfeld et. al. in Asiacrypt 2003. This notion allows a signature holder to designate the signature to a desired designated-verifier. In this paper, we extend this notion to allow a signature holder to designate the signature to multi verifiers, and hence, we call our scheme as Universal Designated Multi Verifier Signatures. We provide security proofs for our schemes based on the random oracle model.

Keywords

universal designated-verifier signature scheme, signature holder, multi verifier, random oracle model

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Ng, CY, Susilo W & Mu, Y, Universal designated multi verifier signature schemes, Proceedings, 11th International Conference on Parallel and Distributed Systems (ICPADS'05), July 2005, 2, 305-309. Copyright IEEE 2005.

Universal Designated Multi Verifier Signature Schemes

Ching Yu Ng, Willy Susilo and Yi Mu
Centre for Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong, NSW 2522, Australia
Email: {cyn27, wsusilo, ymu}@uow.edu.au

Abstract

The notion of Universal Designated-Verifier Signatures was put forth by Steinfeld et. al. in Asiacrypt 2003. This notion allows a signature holder to designate the signature to a desired designated-verifier. In this paper, we extend this notion to allow a signature holder to designate the signature to multi verifiers, and hence, we call our scheme as Universal Designated Multi Verifier Signatures. We provide security proofs for our schemes based on the random oracle model.

1. Introduction

Due to the abundance of electronic applications of digital signatures, many additional properties are needed. The notion of undeniable signature was proposed by Chaum and van Antwerpen in 1989 [3]. In this notion, the signature is only verifiable with the signer's consent by engaging interactively or non-interactively in a confirmation or disavowal protocol. In short, undeniable signatures are *not* universally verifiable. This notion is useful in cryptography such as in licensing software and auctions. It was known that this type of signature schemes has some drawbacks due to blackmailing and mafia attacks [6, 5]. To overcome this problem, designated-verifier technique was proposed in [7], by allowing a non-interactive proof provided by the signer. This scheme is known to be the first non-interactive scheme of Chaum's scheme [2]. The idea of constructing designated verifier signature schemes from any bilinear maps was proposed in [8]. In [4], Desmedt raised the problem of generalizing the designated verifier signature concept to a multi designated verifier scheme. This question was answered affirmatively in [9], where a construction of multi designated verifiers signature scheme was proposed.

Motivated by privacy issues associated with dissemination of signed digital certificates, Steinfeld *et. al.* proposed the notion of *Universal Designated-Verifier Signature* (UDVS) schemes [10]. In this notion, a signature holder can designate the signature to any desired designated verifier, using the verifier's public key. They also showed that bilinear maps allow an elegant construction of a UDVS scheme. An efficient extension of standard RSA/Schnorr signature schemes to UDVS schemes was proposed in [11].

1.1. Our contributions

We extend the notion of UDVS schemes to *Universal Designated Multi Verifier Signature* (UDMVS) schemes. In the new notion, a signature holder can designate the signature to a group of designated verifiers. We present two schemes that fit into our model. We show an efficient construction of UDMVS schemes based on bilinear pairing. We provide security proofs for our schemes based on the random oracle model.

1.2. Organization of The Paper

The rest of this paper is organized as follows. In the next section, we will provide some preliminaries and background required in this paper. In section 3, we define the notion of UDMVS schemes. In section 4, we present a concrete UDMVS scheme based on the bilinear pairing. Section 5 concludes the paper.

2. Preliminaries

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

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE

1. *Bilinearity*: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ 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 $\hat{e}(P, Q) \neq 1$.
3. *Computability*: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ 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 $\mathbb{G}_1 \neq \mathbb{G}_2$.

Bilinear pairing instance generator is defined as a probabilistic polynomial time algorithm $\mathcal{IG}$ that takes as input a security parameter ℓ and returns a uniformly random tuple $param = (p, \mathbb{G}_1, \mathbb{G}_M, \hat{e}, P)$ of bilinear parameters, including a prime number p of size ℓ, a cyclic additive group $\mathbb{G}_1$ of order q, a multiplicative group $\mathbb{G}_M$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_M$ and a generator P of $\mathbb{G}_1$. For a group $\mathbb{G}$ of prime order, we denote the set $\mathbb{G}^* = \mathbb{G} \setminus \{\mathcal{O}\}$ where $\mathcal{O}$ is the identity element of the group.

2.2. Complexity Assumptions

Definition 1. Bilinear Diffie-Hellman (BDH) Problem. *Given 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 $\hat{e}(P, P)^{abc}$.*

For the BDH problem to be hard, $\mathbb{G}_1$ and $\mathbb{G}_M$ must be chosen so that there is no known algorithm for efficiently solving the Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_M$. We note that if the BDH problem is hard for a pairing $\hat{e}$, then it follows that $\hat{e}$ is non-degenerate.

Definition 2. BDH Assumption. *If $\mathcal{IG}$ is a BDH parameter generator, the advantage $\mathtt{Adv}_{\mathcal{IG}}(\mathcal{A})$ that an algorithm $\mathcal{A}$ has in solving the BDH problem is defined to be the probability that the algorithm $\mathcal{A}$ outputs $\hat{e}(P, P)^{abc}$ on inputs $\mathbb{G}_1, \mathbb{G}_M, \hat{e}, P, aP, bP, cP$, where $(\mathbb{G}_1, \mathbb{G}_M, \hat{e})$ is the output of $\mathcal{IG}$ for sufficiently large security parameter ℓ, P is a random generator of $\mathbb{G}_1$ and a, b, c are random elements of $\mathbb{Z}_q$. The BDH assumption is that $\mathtt{Adv}_{\mathcal{IG}}(\mathcal{A})$ is negligible for all efficient algorithms $\mathcal{A}$.*

3. Universal Designated Multi Verifier Signature Scheme

The definition of a Universal Designated Multi Verifier Signature (UDMVS) scheme is very similar to a UDVS scheme. A UDMVS is a tuple of seven algorithms (that may be randomized) as follows:

1. **Common Parameter Generation** (`Setup`): is an algorithm that accepts a security parameter k and outputs a string consisting of common scheme parameters that are publicly shared by all users, cp.
2. **Signer Key Generation** (`SKeygen`): is an algorithm that accepts a common scheme parameter cp and outputs a secret/public key pair (sk_S, pk_S).
3. **Verifiers Key Generation** (`VKeygen`): is an algorithm that accepts a common scheme parameter cp and a number n as the number of verifiers, outputs n secret/public key pairs (sk_i, pk_i), $i = 1, \cdots, n$, for n verifiers.
4. **Signature Generation** (`Sign`): is an algorithm that accepts a common scheme parameter cp, a signer's secret key sk_S and a message m, outputs signer's publicly verifiable signature σ.
5. **Public Verification** (`Verify`): is an algorithm that accepts a common scheme parameter cp, a signer's public key pk_S, a message m and a signature σ, outputs `True` if the verification is correct or $\perp$ otherwise.
6. **Designation** (`Designate`): is an algorithm that accepts a common scheme parameter cp, a signer's public key pk_S, verifiers' public key $pk_1, \cdots, pk_n$ and a message/signature pair (m, σ), outputs a designated multi verifier signature $\hat{\sigma}$.
7. **Designated Verification** (`DVerify`): is an algorithm that accepts a common scheme parameter cp, a signer's public key pk_S, verifiers' secret key sk_i, $i = 1, \cdots, n$ and a message/designated multi verifier signature pair $(m, \hat{\sigma})$, outputs `True` if the verification is correct or $\perp$ otherwise.

3.1. Security Notions

An UDMVS scheme should satisfy the following security properties.

Completeness. We require UDMVS scheme to satisfy the following probability equation:

$$Pr[\mathtt{DVerify}(cp, pk_S, sk_1, \cdots, sk_n, m, \hat{\sigma})] = 1$$

where $cp \leftarrow \mathtt{Setup}(k)$, $(sk_S, pk_S) \leftarrow \mathtt{SKeygen}(cp)$, $(sk_i, pk_i) \leftarrow \mathtt{VKeygen}(cp, n)$, $\sigma \leftarrow \mathtt{Sign}(cp, sk_S, m)$, $\mathtt{True} \leftarrow \mathtt{Verify}(cp, pk_S, m, \sigma)$, $\hat{\sigma} \leftarrow \mathtt{Designate}(cp, pk_S, pk_1, \cdots, pk_n, m, \sigma)$

Non-Transferability. We require a UDMVS scheme to be *non-transferable*. The non-transferability property is ensured by a `transcript simulation` algorithm that can be performed by all designated verifiers to produce an indistinguishable signature from the one that should be produced by the signature holder.

Unforgeability. We provide a formal definition of existential unforgeability of a UDMVS scheme under a chosen message attack (UF-CMA). It is defined using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- Let $\mathcal{A}$ be the UF-CMA adversary. In the startup of the game, $\mathcal{C}$ provides the common scheme parameter, cp, to $\mathcal{A}$, where $cp \leftarrow \texttt{Setup}(k)$ and k is the security parameter.
- $\mathcal{C}$ provides the signer's public key pk_S and verifiers' public key $pk_1, \cdots, pk_n$ to $\mathcal{A}$.
- At any time, $\mathcal{A}$ can query the hash oracle for the hash result on any message m_i of his choice up to q_H times (which is polynomial in k). $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the hash value $H(m_i)$.
- At any time, $\mathcal{A}$ can query the signing oracle for the signature on any message m_i of his choice specifying any user he likes up to q_S times (which is polynomial in k). $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the value $\sigma = \texttt{Sign}(cp, sk_S, m_i)$ where sk_S is the corresponding secret key of the specified user queried by $\mathcal{A}$ for m_i.
- $\mathcal{C}$ will not answer any `Verify` request because $\mathcal{A}$ can verify the signature by himself.
- Eventually, $\mathcal{A}$ will output a valid UDMVS for a message m^* that has never been queried to the signing oracle before, for the designated verifiers with public keys $pk_1, \cdots, pk_n$.

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

$$Succ_{\mathcal{A}}^{UF-UDMVS-CMA}(k)$$

Definition 3. UF-CMA Secure *We say that a UDMVS 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. In other words,*

$$Succ_{\mathcal{A}}^{UF-UDMVS-CMA}(k) \leq \epsilon$$

4. A Concrete UDMVS Scheme from Bilinear Pairing

4.1. A Trivial Scheme

We start our concrete UDMVS scheme by modifying the UDVS scheme proposed in [10]. We note that this scheme can be trivially modified to achieve a UDMVS scheme as follows:

- `Setup`: Select a bilinear group-pair $(\mathbb{G}_1, \mathbb{G}_M)$ of prime order q, where $q = |\mathbb{G}_1| = |\mathbb{G}_M|$ with description string $D_{\mathbb{G}}$ specifying a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_M$ and a generator $P \in \mathbb{G}_1$, together with a cryptographic hash function $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$. The common scheme parameter is $cp = (D_{\mathbb{G}}, P, H_0)$.
- `SKeygen`: Given cp, pick a random $sk_S \overset{R}{\in} \mathbb{Z}_q^*$ and compute $pk_S = sk_S P$. Let $P_{pub} = pk_S$, the signer's public key is (cp, P_{pub}) and the private key is (cp, sk_S).
- `VKeygen`: Given cp and n as the number of the verifiers, pick a different random $sk_i \overset{R}{\in} \mathbb{Z}_q^*$, $i = 1, \cdots, n$ for each verifier and compute $pk_i = sk_i P$. The public key for verifier i is (cp, pk_i) and the private key is (cp, sk_i).
- `Sign`: Given the signer's secret key (cp, sk_S), and a message m, compute $\sigma = sk_S H_0(m)$ as the signature on m.
- `Verify`: Given the signer's public key (cp, P_{pub}) and a message/signature pair (m, σ), accept the signature iff $\hat{e}(P, \sigma) \overset{?}{=} \hat{e}(P_{pub}, H_0(m))$ holds with equality. Otherwise, return $\perp$.
- `Designate`: Given a set of verifiers' public key $(cp, pk_1, \cdots, pk_n)$ and a message/signature pair (m, σ), compute $\hat{\sigma_i} = \hat{e}(pk_i, \sigma)$, for all $i = 1, \cdots, n$.
- `DVerify`: Given a signer's public key (cp, pk_S), a set of verifiers' secret/public key $(cp, (sk_1, pk_1), \cdots, (sk_n, pk_n))$ and a set of message/designated verifier signatures $(m, \hat{\sigma_1}, \cdots, \hat{\sigma_n})$, accept iff $\hat{\sigma_i} \overset{?}{=} \hat{e}(P_{pub}, H_0(m))^{sk_i}$, $i = 1, \cdots, n$, holds with equality. Otherwise, return $\perp$.

Correctness. The correctness of the trivial UDMVS scheme is justified as follows: $\hat{\sigma_i} \overset{?}{=} \hat{e}(P_{pub}, H_0(m))^{sk_i} = \hat{e}(sk_S P, H_0(m))^{sk_i} = \hat{e}(sk_i P, sk_S H_0(m)) = \hat{e}(pk_i, \sigma)$. □

Non-Transferability. The non-transferability is achieved because each verifier can simulate the signature $\hat{\sigma_i}$ by producing an indistinguishable signature $\hat{\sigma_i}'$ from the one that was designated by a signature holder as follows: $\hat{\sigma_i}' = \hat{e}(P_{pub}, H_0(m))^{sk_i} = \hat{\sigma_i}$ We note that since P_{pub} is publicly available, any verifier can produce such a signature using his own private key (cp, sk_i).

Efficiency. The signature produced by the `Designate` algorithm is of the form $(\hat{\sigma_1}, \cdots, \hat{\sigma_n})$, which results in an $n|\mathbb{G}_1|$ bits signature. In the next section, we will show an efficient construction of UDMVS scheme that only requires $|\mathbb{G}_1|$ bit length. Since this construction is trivial, we omit the formal definition of existential unforgeability for this scheme.

4.2. An Efficient UDMVS Scheme

In this section, we present an efficient UDMVS scheme from bilinear pairing. The new scheme is more efficient compared to the scheme that we presented earlier. However, the model is quite different from the one that we used above. In this scheme, the verification requires a collaboration of all of the designated verifiers (in contrast to the previous construction where each verifier can verify by himself/herself). This leads to a very efficient scheme, that only requires $|\mathbb{G}_1|$ bit signature. The construction is as follows:

- `Setup, SKeygen, VKeygen, Sign, Verify`: The same as our trivial scheme.
- `Designate`: Given a set of verifiers' public key $(cp, pk_1, \cdots, pk_n)$ and a message/signature pair (m, σ), compute $\hat{\sigma} = \hat{e}(\sigma, \sum_{i=1}^{n} pk_i)$
- `DVerify`: Given a signer's public key (cp, P_{pub}), a set of verifiers' secret/public key $(cp, (sk_1, pk_1), \cdots, (sk_n, pk_n))$ and a message/designated multi verifier signature pair $(m, \hat{\sigma})$, each verifier performs the following algorithm:
  - Sign the message m as $\widetilde{\sigma_i} = sk_i H_0(m)$ and publish it among the n verifiers.
  - Run `Verify`$(cp, pk_j, m, \widetilde{\sigma_j}), j = 1, \cdots, n$ to validate all the σ_j received. Fail if $\perp$ is returned in any one of the signatures.
  - Test whether $\hat{\sigma} \stackrel{?}{=} \prod_{i=1}^{n} \hat{e}(\widetilde{\sigma_i}, P_{pub})$ holds with equality. Return `true` if it holds, or $\perp$ otherwise.

Correctness. The correctness of the `DVerify` algorithm is justified as follows: $\hat{\sigma} \stackrel{?}{=} \prod_{i=1}^{n} \hat{e}(\widetilde{\sigma_i}, P_{pub}) = \prod_{i=1}^{n} \hat{e}(sk_i H_0(m), sk_S P) = \prod_{i=1}^{n} \hat{e}(sk_S H_0(m), sk_i P) = \prod_{i=1}^{n} \hat{e}(\sigma, sk_i P) = \hat{e}(\sigma, \sum_{i=1}^{n} sk_i P) = \hat{e}(\sigma, \sum_{i=1}^{n} pk_i)$. □

Non-Transferability. Let n verifiers collude to generate a signature on a message m. Each of them will perform the following:

- Sign the message as $\sigma_i = sk_i H_0(m)$ and send it to the other verifiers.
- Check if all of the $\sigma_j, j = 1, \cdots, n$ received are valid, if no, then fail.
- Compute $\acute{\sigma} = \prod_{i=1}^{n} \hat{e}(\sigma_i, P_{pub})$

Note that $\acute{\sigma}$ is indistinguishable from the signature $\hat{\sigma}$ that should have been generated by a signature holder. Hence, no other third party will be convinced with the authenticity of the signature. However, a user in the verifiers group will be convinced because if he/she has not colluded, then he/she is ensured that the signature is authentic.

Unforgeability. Let $\mathcal{A}$ be a UF-CMA adversary in the unforgeability game. We will build a simulator $\mathcal{B}$ that will use $\mathcal{A}$ to solve an instance of the BDH problem. The purpose of the algorithm $\mathcal{B}$ is to compute $\hat{e}(P, P)^{abc}$ from (P, aP, bP, cP) for unknown a, b, c, which is given in the beginning of the game. The simulation is modified from [1] and is as follows:

- $\mathcal{B}$ provides $\mathcal{A}$ the common scheme parameter cp and sends aP as the public key P_{pub} of the signer to $\mathcal{A}$.
- $\mathcal{B}$ generates some random numbers $u_i \stackrel{R}{\in} \mathbb{Z}_q^*$ and computes $u_i cP$ as the public keys $pk_i, i = 1, \cdots, n$ of the verifiers and gives them to $\mathcal{A}$.
- Every time when $\mathcal{A}$ issues a hash query on any message $m_i, i = 1, \cdots, q_H$ of his choice, $\mathcal{B}$ will answer the query as follows.
  - $\mathcal{B}$ maintains a hash record $[m, H(m), r, f]$ to store all the hash results, it grows as new hash result has replied.
  - If the query on m_i has not been asked before (and hence, it does not exist in the record maintained by $\mathcal{B}$), then $\mathcal{B}$ picks a random number $r_i \stackrel{R}{\in} \mathbb{Z}_q^*$ and flips a $\{0, 1\}$ coin that has probability α on outcome 0 and $1 - \alpha$ on outcome 1. If 0 is obtained, $\mathcal{B}$ answers with $H(m_i) = r_i P$. Otherwise, $\mathcal{B}$ answers with $H(m_i) = bP + r_i P$. $\mathcal{B}$ updates his record with $(m_i, H(m_i), r_i, f_i)$, where $f_i \in \{0, 1\}$ is the result of the coin flipping.
  - If the query on m_i has been asked before, then $\mathcal{B}$ looks up his record to obtain the entry $(m_i, H(m_i), r_i, f_i)$ and answers with the stored value $H(m_i)$.
- Every time when $\mathcal{A}$ issues a sign query on any message $m_i, i = 1, \cdots, q_S$ and any public key $pk_j, j = 1, \cdots, n$ of his choice, $\mathcal{B}$ will answer the query as follows:
  - If the query on (m_i, pk_j) has not been asked before (and hence, it does not exist in the record maintained by $\mathcal{B}$), then $\mathcal{B}$ picks a random number $r_i \stackrel{R}{\in} \mathbb{Z}_q^*$ and answers the query with $r_i pk_j$. $\mathcal{B}$ updates his record with $(m_i, r_i P, r_i, 0)$. Note that $r_i pk_j = r_i sk_j P = sk_j r_i P = sk_j H(m_i)$, which is equal to the signature on m_i signed with the private key corresponds to pk_j.
  - If the query on (m_i, pk_j) has been asked before, then $\mathcal{B}$ looks up on his record to find the entry $(m_i, H(m_i), r_i, f_i)$. If f_i is found to be equal to 1 (i.e. $H(m_i) = bP + r_i P$), then $\mathcal{B}$ terminates and fails the simulation. Otherwise if f_i is found to be equal to 0 (i.e. $H(m_i) = r_i P$), then $\mathcal{B}$ return $r_i P$ as the answer.

- Eventually, $\mathcal{A}$ will output a forged UDMVS pair (m^*, σ^*) designated to all the verifiers on a message m^* that seems to have been signed by the signer. $\mathcal{B}$ needs to look up on his record to find the entry $(m_i, H(m_i), r_i, f_i)$ where $m_i = m^*$. If m^* has not been queried (i.e. the entry is not found), then $\mathcal{B}$ terminates the game with failure. But since the random values r_i are randomly picked over $\mathbb{Z}_q^*$, thanks to its uniform randomness, the hash results are distributed over $\mathbb{G}_1$ and the probability that $\mathcal{A}$ hits the hash result $H(m^*)$ is $\frac{1}{q}$ where q is a large prime, which is negligible. Hence m^* must have been queried during the hash queries (i.e. $\mathcal{A}$ has obtained $H(m^*)$ from $\mathcal{B}$) and $\mathcal{B}$ is able to find the entry $(m^*, H(m^*), r_{i^*}, f_{i^*})$ where i^* denotes the index where $m_{i^*} = m^*$. In order to compute the answer for the given instance to the BDH problem, $H(m^*)$ has to be in the form of $bP + r_{i^*}P$ (i.e. $f_{i^*} = 1$). If it is not, $\mathcal{B}$ terminates and fails the simulation. Otherwise, $\mathcal{B}$ calculates and outputs $\sigma^{*(\sum_{i=1}^{n} u_i)^{-1}} \cdot \hat{e}(aP, cP)^{-r_{i^*}}$.

If $\mathcal{B}$ does not terminate in the simulation, the answer computed by $\mathcal{B}$ is equal to :

$$\begin{aligned}
& \sigma^{*(\sum_{i=1}^{n} u_i)^{-1}} \cdot \hat{e}(aP, cP)^{-r_{i^*}} \\
= \; & \hat{e}(\sigma, \textstyle\sum_{i=1}^{n} pk_i)^{(\sum_{i=1}^{n} u_i)^{-1}} \cdot \hat{e}(aP, cP)^{-r_{i^*}} \\
= \; & \hat{e}(aH(m^*), \textstyle\sum_{i=1}^{n} u_i cP)^{(\sum_{i=1}^{n} u_i)^{-1}} \cdot \hat{e}(aP, cP)^{-r_{i^*}} \\
= \; & \hat{e}(a(bP + r_{i^*}P), cP) \cdot \hat{e}(aP, cP)^{-r_{i^*}} \\
= \; & \hat{e}(abP, cP) \cdot \hat{e}(ar_{i^*}P, cP) \cdot \hat{e}(-ar_{i^*}P, cP) \\
= \; & \hat{e}(P, P)^{abc}
\end{aligned}$$

Hence $\mathcal{B}$ has successfully solved the BDH problem for the given instance (P, aP, bP, cP). Let $\beta = Succ_{\mathcal{A}}^{UF-UDMVS-CMA}(k)$, the probability that $\mathcal{B}$ successes is:

$$Pr[f_i = 0, i = 1, \cdots, q_S] \times Pr[f_{i^*} = 1] \times \beta = \alpha^{q_S}(1-\alpha)\beta$$

In order to have a maximum probability of success, we take derivative on this value and found it is maximize at $\alpha = q_S(q_S + 1)^{-1}$. Hence $\mathcal{B}$ solves the BDH problem with probability:

$$\begin{aligned}
& (q_S(q_S+1)^{-1})^{q_S}(1 - q_S(q_S+1)^{-1})\beta \\
= \; & (\tfrac{q_S}{q_S+1})^{q_S}(\tfrac{\beta}{q_S+1}) \\
= \; & (1 + \tfrac{1}{q_S})^{-q_S}(\tfrac{\beta}{q_S+1}) \\
\geq \; & \tfrac{\beta}{e(q_S+1)}
\end{aligned}$$

where e is the base for natural logarithm. In other words, $\mathcal{B}$ solved the BDH problem with non-negligible probability, which contradicts with the BDH assumption. Therefore, we complete the proof.

Efficiency. The signature produced by our UDMVS scheme is $|\mathbb{G}_1|$ bit length, which is very efficient.

5. Conclusion

In this paper, we firstly proposed the notion of Universal Designated Multi Verifier Signature (UDMVS) schemes. We formalized this notion by proposing their model and security requirements. We proceeded with an efficient construction of UDMVS scheme based on bilinear pairing that only requires $|\mathbb{G}_1|$ bit length signature and provided a formal security proof. Furthermore, we note that if we combine the `Sign` and `Designate` algorithms in our efficient UDMVS scheme, we will obtain a designated multi verifier scheme, which turns out to be more efficient than the construction proposed in [9].

References

[1] D. Boneh and M. Franklin. Identity-based encryption from the weil pairings. In *Advances in Cryptology-Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[2] D. Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology - Eurocrypt '90*, pages 458–464, 1990.

[3] D. Chaum and H. van Antwerpen. Undeniable signatures. In *Advances in Cryptology - Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216, 1990.

[4] Y. Desmedt. Verifier-designated signatures. In *Rump Session, Crypto 2003*, 2003.

[5] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the fiat-shamir passport protocol. In *Advances in Cryptology - Crypto 87*, pages 21–39, 1998.

[6] Y. Desmedt and M. Yung. Weaknesses of Undeniable Signature Schemes. In *Advances in Cryptology - Eurocrypt '91*, volume 547 of *Lecture Notes in Computer Science*, pages 205–220, 1991.

[7] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. In *Advances in Cryptology - Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154, 1996.

[8] F. Laguillaumie and D. Vergnaud. Designated Verifiers Signature: Anonymity and Efficient Construction from Bilinear Map. In *Fourth Conference on Security in Communication Networks '04 (SCN04)*, Lecture Notes in Computer Science, 2004.

[9] F. Laguillaumie and D. Vergnaud. Multi-Designated Verifiers Signatures. In *Sixth International Conference on Information and Communications Security (ICICS 2004)*, Lecture Notes in Computer Science, 2004(to appear).

[10] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In *Proceedings of Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–543, 2003.

[11] R. Steinfeld, H. Wang, and J. Pieprzyk. Efficient Extension of Standard Schnorr/RSA signatures into Universal Designated-Verifier Signatures. In *Proceedings of 7th International Workshop on Theory and Practice in Public Key Cryptography (PKC 2004)*, volume 2947 of *Lecture Notes in Computer Science*, pages 86–100, 2004.