

# Universal Designated Verifier Signature Proof (or How to Efficiently Prove Knowledge of a Signature)

Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo

Centre for Information Security,  
School of Information Technology and Computer Science  
University of Wollongong  
Wollongong, NSW 2519, Australia  
{baek, rei, wsusilo}@uow.edu.au

**Abstract.** Proving knowledge of a signature has many interesting applications. As one of them, the Universal Designated Verifier Signature (UDVS), introduced by Steinfeld et al. in Asiacrypt 2003 aims to protect a signature holder’s privacy by allowing him to convince a verifier that he holds a valid signature from the signer without revealing the signature itself. The essence of the UDVS is a transformation from a publicly verifiable signature to a designated verifier signature, which is performed by the signature holder who does *not* have access to the signer’s secret key. However, one significant inconvenience of all the previous UDVS schemes considered in the literature is that they require the designated verifier to create a public key using the signer’s public key parameter and have it certified to ensure the resulting public key is compatible with the setting that the signer provided. This restriction is unrealistic in several situations where the verifier is not willing to go through such setup process. In this paper, we resolve this problem by introducing a new type of UDVS. Different from previous approach to UDVS, our new UDVS solution, which we call “Universal Designated Verifier Signature Proof (UDVSP)”, employs an *interactive protocol* between the signature holder and the verifier *while maintaining high level of efficiency*. We provide a formal model and security notions for UDVSP and give two constructions based on the bilinear pairings. We prove that the first construction is secure in the random oracle model and so is the second one in the standard model.

## 1 Introduction

### 1.1 Motivation

Consider a situation where Alice, who has graduated from the University ABC, would like to apply for a job online. In this situation, Alice has to convince the employer that she indeed holds the diploma that has been signed by the registrar of the University ABC. However, she does not want to send her diploma away, since she feels that anyone else might be able to use it for a different purpose.

A normal digital signature cannot satisfy her requirement as anyone who has obtained a message (in the above example, Alice’s diploma) and a signature on it (the registrar’s signature on Alice’s diploma) can easily *copy* them. For this reason, Alice seeks a new type of digital signature that can provide a property that the signature on Alice’s diploma is *non-transferable* meaning that Alice can convince Bob that she is in possession of a valid diploma awarded by the University ABC but having talked to Alice, Bob cannot convince any other parties about the truth of the statement.

At a glance, it seems that Alice’s need can be met by the Universal Designated Verifier Signature (UDVS), introduced By Steinfeld et al. in Asiacrypt 2003 [25]. In UDVS, a “designator (signature holder)” who has obtained a valid signature from a “signer” can convince a “designated verifier” that he holds a valid signature obtained from the signer but other parties including the designated verifier himself cannot convince other parties of the validity of the same statement. In spite of their elegant structures, the difficulty of adopting the UDVS schemes proposed so far including those from [25, 26, 28] to the Alice’s diploma verification problem is that they require designated verifiers to create private/public key pairs *using the same public key parameter that has been set by the signer* and have them certified. (Indeed, the protocol for conducting such a task, “Verifier Key-Registration”, is included as a sub-algorithm of UDVS [25]). This is sometimes hard to be realized especially when proving knowledge of a signature obtained from the original signer is only a designator’s interest. – In the above scenario, for example, the employer will be less likely to agree on creating a private/public key pair according to the public parameter set by Alice’s university just only to verify her diploma, as this key setup involving management of Public Key Infrastructure (PKI) may incur significant cost. Hence, although the UDVS seems to be a good candidate to solve Alice’s diploma verification problem, the public key setup requirement for verifiers remains as a problem.

Motivated by the above problem of the UDVS schemes considered in the previous literature, we would like to obtain a new type of UDVS that eliminates the assumption of having the verifier generate a private/public key pair to verify the signature holder’s claim. In order to achieve this, the natural methodology one can envision is to adopt the *interactive* proof [15] style protocol. However, interactive protocols sometimes significantly degrade the system efficiency. So another important requirement of the new UDVS is *high level of efficiency*. Our new UDVS proposed in this paper will satisfy both requirements.

## 1.2 Our Contributions

In this paper, we propose a new type of UDVS, which we call “Universal Designated Verifier Signature Proof (UDVSP)” and provide two highly efficient constructions based on the bilinear pairings [4, 18]. Informally, our UDVSP system has the following characteristics: 1) The signer signs a message and provides the message/signature pair to the designator (signature holder); 2) the designator *transforms* the signer’s signature and, using an interactive protocol that takes this transformed signature as common input, convinces the designated verifier

that he has indeed obtained the *valid* signature on the message from the signer; and 3) the designated verifier does not have to setup a private/public key.

In terms of security, we formalize the security of UDVSP against *impersonation* attack: The system should not allow the attackers including the designated verifier to convince any other person that the designator indeed holds a valid signature from the signer. Our two concrete constructions of UDVSP are proven to be secure under the proposed security model relative to known computational assumptions related to hardness of the discrete-logarithm problem. The security proof for the first construction needs the random oracle model [2] while the second one does not depend on this assumption.

Our two constructions of UDVSP are highly efficient. The signing algorithms of them are the same as those of underlying signature schemes [5, 7], which are very efficient. The transformation algorithms just need one exponentiation and the interactive verification protocols only require two bilinear pairing operations on top of the Schnorr identification protocol [23]. (Note that the details of these algorithms and protocol will be given in Section 3).

We remark that throughout this paper, the term “designated” has a literal meaning that a *specified* verifier participates in the protocol run by the signature holder. This does not mean that it provides *explicit* authentication of the verifier’s identity, which is provided in the non-interactive designated signature [17] where the verifiers are authenticated by checking the certificate of their public keys. Note that as mentioned in [17], the advantage of the non-interactive designated verifier signature is that it can prevent somewhat strong attacks such as mafia (a.k.a. man-in-the-middle) attacks [12] and blackmailing [16] as the verifiers are explicitly authenticated by their public keys (which come with the certificates issued by the trusted third party).

### 1.3 Related Work

The concept of UDVS was proposed by Steinfeld et al. [25], who constructed a concrete scheme that realizes the concept using the bilinear pairings. They also observed that Boneh et al.’s [6] ring signature [24] converted to two-signer ring signature can be viewed as UDVS. The UDVS schemes based on the RSA and Schnorr signatures were soon followed [26]. Very recently, the UDVS scheme based on the bilinear pairings whose security can be analyzed without the random oracle assumption was proposed by Zhang et al. [28]. Although these schemes are elegant, as discussed earlier, the assumption of having designated verifiers to generate their own private/public key pairs according to the signer’s public key parameter setting is strong and even unrealistic in some situations. We note that the reason why one needs such a key registration process for verifiers in the UDVS schemes is that *non-transferability* of a signature obtained from the original signer is achieved via the technique of trapdoor commitment [8] non-interactively as widely adopted in the various designated-verifier signature schemes [17, 19, 27].

In contrast, undeniable signature proposed by Chaum and Antwerpen [9] achieves non-transferability (of the original signer’s signature) via an interactive

protocol although non-interactive version of the protocol is also possible using the trapdoor commitment. (We note that the security analysis of Chaum’s undeniable signature is given only recently by Ogata et al. [22]). However, as discussed in [25], the crucial difference between the UDVS and the undeniable signature (including other designated-verifier signatures) is that in the latter, only the *signer* who possesses the secret signing key can designate a signature while in the former, *any* legitimate user who possesses the signature from the signer can perform designation.

Another related area of UDVS is anonymous credential system [20, 9, 10]. However, as mentioned in [25], this area of research focuses more on user privacy such as “selective disclosure” of attributes and “unlinkability” of user transaction. Our work (and the work related to UDVS [25, 26, 28] in general) focuses more on providing an *efficient* mechanism to convince designated verifiers about the truth of the statement that a signature holder is in possession of a valid signature from the original signer. Consequently, we can avoid *heavy* zero knowledge proof protocols used in many credential systems such as [20, 9, 10]. Interestingly, our first construction of UDVSP shows that if proving knowledge of a signature is only concern, one can simply use Boneh, Lynn and Shacham (BLS)’s [5] pairing-based signature scheme whose security against chosen message attack is relative to the *standard* Computational Diffie-Hellman (CDH) problem, as opposed to the one used in [20], whose security against chosen message attack is based on the fairly complex and non-standard assumption called “LRSW assumption”.

Moreover, in order to precisely establish the end goal of security of our UDVSP systems, the security analysis given in this paper does not depend on the auxiliary properties such as proof of knowledge [1], witness indistinguishability [13] and honest verifier zero knowledge and so on.

Finally, we remark that Naor [21] also pointed out that it is not desirable to assume that the verifier of the authentication is part of the system and has established a public key. This is because it is often difficult to assure the independence of keys in the PKI and there is no reason to assume that the verifier has chosen his key properly.

## 2 Preliminaries

### 2.1 Symbols and Notations

We use the notation  $A(\cdot, \dots, \cdot)$  to denote an algorithm, with input arguments separated by commas. (Note that our underlying computational model is a probabilistic Turing Machine). The notation  $A^{O(\cdot)}(\cdot, \dots, \cdot)$  denotes that algorithm  $A$  makes calls to an oracle  $O(\cdot)$ . We use  $a \leftarrow A(x_1, \dots, x_n)$  to denote the assignment of a uniformly and independently distributed random element from the output of  $A$  on input  $(x_1, \dots, x_n)$  to the variable  $a$ .

We use the notation  $\text{Prot}[P(\underbrace{\cdot, \dots, \cdot}_{\text{private input}}) \leftrightarrow V(\underbrace{\cdot, \dots, \cdot}_{\text{private input}})(\underbrace{\cdot, \dots, \cdot}_{\text{common input}})]$  to denote an interactive protocol  $\text{Prot}$  between a prover  $P$  and a verifier  $V$ , both

of which are modeled as probabilistic Turing Machines. *Private* inputs for  $P$  and  $V$  are presented inside the parentheses immediately followed by the letters “ $P$ ” and “ $V$ ”. *Common* inputs for  $P$  and  $V$  are presented inside the parentheses followed by the square brackets “[ ]”.

Given a set  $S$ , we denote by  $b \stackrel{R}{\leftarrow} S$  the assignment of a uniformly and independently distributed random element from the set  $S$  to the variable  $b$ .

We say a probability function  $f : \mathbb{N} \rightarrow \mathbb{R}_{[0,1]}$  is negligible in  $k$  if, for all  $c > 0$ , there exists  $k_0 \in \mathbb{N}$  such that  $f(k) \leq \frac{1}{k^c}$  whenever  $k \geq k_0$ . Here,  $\mathbb{R}_{[0,1]} = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$ .

## 2.2 Computational Primitives

The bilinear pairing  $e$  that will be used throughout this paper is the *admissible* bilinear pairing [4, 18], which is defined over two groups of the same prime-order  $q$  denoted by  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . (By  $\mathbb{G}_1^*$  and  $\mathbb{Z}_q^*$ , we denote  $\mathbb{G}_1 \setminus \{1\}$  where 1 is the identity element of  $\mathbb{G}_1$ , and  $\mathbb{Z}_q \setminus \{0\}$  respectively.) Suppose that  $\mathbb{G}_1$  is generated by  $g$ . Then,  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  has the following properties: 1) Bilinear:  $e(g^a, g^b) = e(g, g)^{ab}$ , for all  $a, b \in \mathbb{Z}_q$  and 2) Non-degenerate:  $e(g, g) \neq 1$ .

For convenience, we define an atomic bilinear pairing parameter generation algorithm that will be used in many parts of the paper.

- $\text{PramGen}_a(k)$ : This algorithm generates two groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$  of order  $q > 2^k$  and creates bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ . It also chooses a generator  $g$  of the group  $\mathbb{G}_1$ . It outputs a parameter  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2)$ .

We now review the definition of “One More Discrete-Logarithm (OMDL)” problem used to analyze the security of the Schnorr identification protocol [23] against impersonation under active attack in which an attacker can participate in the execution of the protocol as a cheating verifier [3].

**Definition 1 (OMDL).** The corresponding experiment for this problem, denoted  $\mathbf{Exp}^{omdl}(k)$ , is defined as follows. Firstly, the atomic parameter generation algorithm is run and parameter  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2)$  is generated. A polynomial-time attacker  $A$  now makes  $n$  queries to the challenge oracle  $\mathcal{C}(\cdot)$  and  $m$  queries to the Discrete-Logarithm (DL) oracle  $\mathcal{DL}_{q,g}(\cdot)$ . Upon receiving a query (null input), the challenge oracle  $\mathcal{C}(\cdot)$  returns a random point  $h \in \mathbb{G}_1$ . Upon receiving a query  $z$ ,  $\mathcal{DL}_{q,g}(\cdot)$  returns  $s$  such that  $g^s = z$ . However, a restriction here is that  $m < n$ . Namely, the number of queries to the DL oracle should be strictly less than the number of queries to the challenge oracle.  $A$ ’s final goal is to output *all* the discrete logarithms of the  $n$  challenges returned from  $\mathcal{C}(\cdot)$ . Formally, we describe  $\mathbf{Exp}^{omdl}(k)$  as follows.

- Experiment:  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2) \leftarrow \text{PramGen}_a(k)$ ;  $(s_1, \dots, s_n) \leftarrow A^{\mathcal{C}(\cdot), \mathcal{DL}_{q,g}(\cdot)}(q, g, e, \mathbb{G}_1, \mathbb{G}_2)$
- Output: If  $(g^{s_1} = h_1) \wedge \dots \wedge (g^{s_n} = h_n)$ , where  $h_1, \dots, h_n$  are random points in  $\mathbb{G}_1$  output by the challenge oracle  $\mathcal{C}(\cdot)$ , and  $m < n$ , where  $m$  denotes the number of queries to the DL oracle, then return 1. Otherwise, return 0.

We define  $A$ 's advantage as  $\mathbf{Adv}_A^{omdl}(k) = \Pr[\mathbf{Exp}^{omdl}(k)] = 1$ . We say that OMDL problem is hard if  $\mathbf{Adv}_A^{omdl}(k)$  is negligible in  $k$ .

Other computational problems that we use in this paper are the Computational Diffie-Hellman (CDH) and the Strong Diffie-Hellman (SDH) problems, which are reviewed in Appendix A.

### 3 Universal Designated Verifier Signature Proof System

#### 3.1 Model

As informally outlined in Section 1, there are three parties involved in Universal Designated Verifier Signature Proof (UDVSP) system: a *signer*, a *designator* (signature holder) and a *designated verifier*. After creating a secret (signing) key and public (verification) key pair, the signer signs a message and transmits the resulting signature together with the message to the designator. It is important that transmission of the signature should be done in a secure manner, e.g. via secure channel. (This is similar to the private key generation process in identity-based cryptography in which a trusted party generates a private key associated with user's identifier information and sends the resulting key via secure channel). Having obtained the valid signature from the signer, the designator creates a transformed signature by generating a random mask and *hiding* the original signature using it. The designator then convinces the designated verifier via an interactive protocol that the transformed signature has been generated from the valid signature obtained from the signer. Below, we formally define UDVSP system.

**Definition 2 (Universal Designated Verifier Signature Proof).** A Universal Designated Verifier Signature Proof system consists of the following five polynomial-time algorithms and a protocol:

- **SigKeyGen:** Taking a security parameter  $k \in \mathbb{N}$  as input, this algorithm generates a signer's public/secret (verification/signing) key pair  $(pk, sk)$ . We write  $(pk, sk) \leftarrow \mathbf{SigKeyGen}(k)$ .
- **Sign:** Taking signer's secret key  $sk$  and a message  $m$  as input, this algorithm generates a signature  $\sigma$  on  $m$ . We write  $\sigma \leftarrow \mathbf{Sign}(sk, m)$ .
- **Verify:** Taking a signer's public key  $pk$ , a signature  $\sigma$  and a message  $m$  as input, this algorithm outputs 1 if  $\sigma$  is a valid signature on  $m$  and outputs 0 otherwise. We write  $d \leftarrow \mathbf{Verify}(pk, \sigma, m)$ , where  $d \in \{0, 1\}$ .
- **Transform:** Taking a signer's public key  $pk$  and a signature  $\sigma$  as input, this algorithm picks a secret mask  $\tilde{sk}$  and generates a transformed signature  $\tilde{\sigma}$  using  $\tilde{sk}$ . It outputs  $\tilde{\sigma}$  and  $\tilde{sk}$ . We write  $(\tilde{\sigma}, \tilde{sk}) \leftarrow \mathbf{Transform}(pk, \sigma)$ .
- **Verify:** This is an interactive verification protocol between a designator and a designated verifier, denoted  $P$  and  $V$  respectively. Common inputs for  $P$  and  $V$  are a signer's public key  $pk$ , a transformed signature  $\tilde{\sigma}$  and a message  $m$ .  $P$ 's private input is a secret mask  $\tilde{sk}$  used to create  $\tilde{\sigma}$ .  $V$  does not have

any private input. In this protocol,  $P$  tries to convince  $V$  that  $\tilde{\sigma}$  has been generated from the valid signature  $\sigma$  obtained from the signer, with the knowledge of  $\tilde{sk}$ . The output of this protocol is verification decision 1 or 0 depending on whether  $V$  accepts or rejects. We write  $d \leftarrow \text{IVerify}[P(\tilde{sk}) \leftrightarrow V](pk, \tilde{\sigma}, m)$ .

Notice that in the above definition,  $\text{SigKeyGen}$  and  $\text{Sign}$  are run by the signer;  $\text{Verify}$  and  $\text{Transform}$  are run by the designator. We emphasize that  $\text{Verify}$  is *not publicly available* in UDVSP system.

We require two *consistency* properties from UDVSP system. One property is that the signature  $\sigma$  on the message  $m$ , produced by the signer should be accepted as valid by the  $\text{Verify}$  algorithm. The other property is that the transformed signature  $\tilde{\sigma}$  produced by the designator using the valid signature  $\sigma$  from the signer and his secret mask  $\tilde{sk}$  should be accepted as valid in the  $\text{IVerify}$  protocol.

### 3.2 Security Notions

The first essential security requirement for UDVSP system is that a signature created by the signer should be existentially unforgeable under (adaptive) chosen message attack, which is a standard requirement for digital signature schemes. Below, we review the formal definition of unforgeability of chosen message attack [14].

**Definition 3 (Unforgeability against Chosen Message Attack).** Suppose that  $\text{SigKeyGen}$ ,  $\text{Sign}$  and  $\text{Verify}$  are as defined in Definition 2. Consider the following experiment  $\text{Exp}^{f\text{-cma}}(k)$  in which a polynomial-time attacker  $A$  after making queries to the signing oracle  $\text{Sign}(sk, \cdot)$ , outputs a *new* message and valid signature pair:

- Experiment:  $(pk, sk) \leftarrow \text{SigKeyGen}(k) : (m, \sigma) \leftarrow A^{\text{Sign}(sk, \cdot)}(pk)$
- Output: If  $1 \leftarrow \text{Verify}(pk, m, \sigma)$  and  $m$  has not been queried to  $\text{Sign}(sk, \cdot)$  then return 1. Otherwise return 0.

We define  $A$ 's advantage in the above experiment as  $\text{Adv}_A^{f\text{-cma}}(k) = \Pr[\text{Exp}^{f\text{-cma}}(k) = 1]$ . We say that the underlying signature scheme of UDVSP system is existentially unforgeable under chosen message attack if  $\text{Adv}_A^{f\text{-cma}}(k)$  is negligible in  $k$ .

The second essential security requirement for UDVSP system is resistance against impersonation attack. That is, UDVSP system should prevent an attacker who does not hold a valid signature created by the signer from impersonating the honest designator who holds a valid signature created by the signer.

We divide this impersonation attack further into two categories, “Type-1” and “Type-2” attacks. In Type-1 attack, an attacker who has obtained a transformed signature participates in the  $\text{IVerify}$  protocol as a *cheating designated verifier* and interacts with an honest designator a number of times. The attacker then tries to impersonate the honest designator to other honest designated verifier. Below, we give a formal definition for security against the Type-1 attack.

**Definition 4 (Security against Impersonation under Type-1 Attack).**

Assume that  $\text{SigKeyGen}$ ,  $\text{Sign}$ ,  $\text{Verify}$ ,  $\text{Transform}$  and  $\text{IVerify}$  are as defined in Definition 2. Suppose that a polynomial-time attacker  $A$  consists of two sub-algorithms  $\hat{V}$  and  $\hat{P}$ , which represent a cheating designated verifier and a cheating designator respectively. Let  $P$  denote an honest designator. Let  $\text{Conv}_{\text{IVerify}}$  be a function that outputs a conversation transcript  $T$  of the  $\text{IVerify}$  protocol between  $P$  and  $\hat{V}$ . Note here that  $T$  is a random variable afreshed by  $P$  and  $\hat{V}$ 's random coins. We write  $T \leftarrow \text{Conv}_{\text{IVerify}}[P(\tilde{sk}) \leftrightarrow \hat{V}](pk, \tilde{\sigma}, m)$ .

Now consider an experiment  $\mathbf{Exp}^{im-type1}(k)$ . Firstly, in this experiment, a signer's public/secret key pair  $(pk, sk)$  is generated using a security parameter  $k \in \mathbb{N}$ .  $pk$  is then given to the honest designator  $P$  and the attacker  $A = (\hat{V}, \hat{P})$ . Next, an arbitrary message  $m$  is chosen and a signature  $\sigma$  on  $m$  is generated. Also, a designator's secret mask  $\tilde{sk}$  is chosen based on  $pk$  and a transformed signature  $\tilde{\sigma}$  is created using  $\tilde{sk}$ .  $\tilde{\sigma}$  is then given to  $A = (\hat{V}, \hat{P})$  while  $\tilde{sk}$  is given *only* to  $P$ .  $\hat{V}$  now interacts with  $P$  in the  $\text{IVerify}$  protocol  $p(k)$  times, where  $p(\cdot)$  denotes a polynomial-time computable function. Having accessed to transcripts of these interactions and  $\hat{V}$ 's random coins used in them, which are denoted  $T_i$  and  $r_i^{\hat{V}}$  respectively for  $i = 1, \dots, p(k)$ ,  $\hat{P}$  tries to *impersonate* the honest designator  $P$  to an honest designated verifier  $V$  in the  $\text{IVerify}$  protocol. Formally,  $\mathbf{Exp}^{im-type1}(k)$  can be described as follows.

- Experiment:  $(pk, sk) \leftarrow \text{SigKeyGen}(k)$ ;  $m \leftarrow \{0, 1\}^*$ ;  
 $\sigma \leftarrow \text{Sign}(sk, m)$ ;  $(\tilde{\sigma}, \tilde{sk}) \leftarrow \text{Transform}(pk, \sigma)$ ;  
 $T_i \leftarrow \text{Conv}_{\text{IVerify}}[P(\tilde{sk}) \leftrightarrow \hat{V}](pk, \tilde{\sigma}, m)$  for  $i = 1, \dots, p(k)$ ;  
 $d \leftarrow \text{IVerify}[\hat{P}((T_1, r_1^{\hat{V}}), \dots, (T_{p(k)}, r_{p(k)}^{\hat{V}})) \leftrightarrow V](pk, \tilde{\sigma}, m)$
- Output:  $d$

We define  $A$ 's advantage in the above experiment as  $\mathbf{Adv}_A^{im-type1}(k) = \Pr[\mathbf{Exp}^{im-type1}(k) = 1]$ . We say that UDVSP system is secure against impersonation under Type-1 attack if  $\mathbf{Adv}_A^{im-type1}(k)$  is negligible in  $k$ .

In Type-2 attack, the attacker simply *ignores the transformed signature that he has obtained before* but tries to create a new transformed signature on his own and use this to impersonate the honest designator to an honest designated verifier in the  $\text{IVerify}$  protocol. In what follows, we formally define the security against Type-2 attack.

**Definition 5 (Security against Impersonation under Type-2 Attack).**

Assume that  $\text{SigKeyGen}$ ,  $\text{Sign}$ ,  $\text{Verify}$ ,  $\text{Transform}$  and  $\text{IVerify}$  are as defined in Definition 2. Let  $A$  be a polynomial-time attacker. Consider an experiment  $\mathbf{Exp}^{im-type2}(k)$ . Firstly, in this experiment, a signer's public/secret key pair  $(pk, sk)$  is generated using a security parameter  $k$  and  $pk$  is given to  $A$ . Then, an arbitrary message  $m$  is chosen and is given to  $A$ .  $A$  then generates a designator's secret mask  $\tilde{sk}'$  and a transformed signature  $\tilde{\sigma}'$  on its own and participates in the  $\text{IVerify}$  protocol with an honest designated verifier  $V$ . Formally,  $\mathbf{Exp}^{im-type2}(k)$  can be described as follows.



- Experiment:  $(pk, sk) \leftarrow \text{SigKeyGen}(k)$ ;  $m \leftarrow \{0, 1\}^*$ ;  $(\tilde{\sigma}', \tilde{sk}') \leftarrow A(pk, m)$ :  
 $d \leftarrow \text{IVerify}[A(\tilde{sk}') \leftrightarrow V](pk, \tilde{\sigma}', m)$
- Output:  $d$

We define  $A$ 's advantage in the above experiment as  $\mathbf{Adv}_A^{im-type2}(k) = \Pr[\mathbf{Exp}^{im-type2}(k) = 1]$ . We say that UDVSP system is secure against impersonation under Type-2 attack if  $\mathbf{Adv}_A^{im-type2}(k)$  is negligible in  $k$ .

## 4 Our Universal Designated Verifier Signature Proof Systems

### 4.1 UDVSP System Based on BLS Signature

Our first UDVSP system is based on Boneh, Lynn and Shacham (BLS)'s [5] signature scheme. Each sub-algorithm and protocol of this system can be described as follows.

- $\text{SigKeyGen}(k)$ : Compute  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2) \leftarrow \text{PramGen}_a(k)$ . Choose  $x \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$  and compute  $y = g^x$ . Specify a hash function  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ . Output  $pk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, H, y)$  and  $sk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, H, x)$ .
- $\text{Sign}(sk, m)$ , where  $m \in \{0, 1\}^*$ : Compute  $\sigma = H(m)^x$ . Output  $\sigma$ .
- $\text{Verify}(pk, m, \sigma)$ : Check  $e(\sigma, g) \stackrel{?}{=} e(H(m), y)$ . If the equality holds, output 1. Otherwise, output 0.
- $\text{Transform}(pk, \sigma)$ : Choose  $z \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$  and compute  $\sigma^z (= H(m)^{xz})$ . Output  $\tilde{\sigma} = \sigma^z$  and  $\tilde{sk} = z$ .
- $\text{IVerify}[P(\tilde{sk}) \leftrightarrow V](pk, \tilde{\sigma}, m)$ :
  - Both  $P$  and  $V$  compute  $v_1 = e(\tilde{\sigma}, g)$ , and  $v_2 = e(H(m), y)$ .
  - 1.  $P$  chooses  $t \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$ , computes  $w = v_2^t$  and sends  $w$  to  $V$ .
  - 2.  $V$  chooses  $c \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$  and sends it to  $P$ .
  - 3.  $P$  computes  $t = s + cz \pmod q$  and sends  $t$  to  $V$ .
  - 4.  $V$  checks  $v_2^t \stackrel{?}{=} wv_1^c$ . If the equality holds then output 1. Otherwise, output 0 otherwise.

If the hash function  $H$  is assumed to be the random oracle [2], one can prove that the  $\text{Sign}$  algorithm above is unforgeable under chosen message attack assuming that the CDH problem (Definition 6) is hard [5] (in  $\mathbb{G}_1$ ).

Notice that the above  $\text{IVerify}$  protocol is a protocol for proving knowledge of  $z$  satisfying the relation  $e(\tilde{\sigma}, g) = e(H(m), y)^z$ . From this,  $P$  can convince the designated verifier that he possesses a valid BLS-signature  $H(m)^x$ . More precisely, if there exists a knowledge extractor [1] that extracts  $z$ , one can use this extractor to construct another knowledge extractor that outputs  $\tilde{\sigma}^{1/z}$  as a valid signature. Indeed, this value is a valid BLS-signature as  $e(\tilde{\sigma}, g) = e(H(m), y)^z$  implies  $e(\tilde{\sigma}, g)^{1/z} = e(H(m), y)$  and hence  $e(\tilde{\sigma}^{1/z}, g) = e(H(m)^x, g)$ . However, as mentioned earlier, we do not use this auxiliary property to analyze the security

of the proposed system as the zero-knowledge or witness-indistinguishability required to provide the security against impersonation under *active* attacks (i.e. attacks other than eavesdropping attack) often compromises the efficiency of protocol [4].

It is important to notice that in the `IVerify` protocol, the designated verifier does not need to setup his private/public key pair, which is a crucial difference compared to previous UDVS schemes.

## 4.2 UDVSP System Based on BB Signature

Our second UDVSP protocol is based on Boneh and Boyen (BB)'s [7] signature scheme. Each sub-algorithm and protocol of this system can be described as follows.

- `SigKeyGen`( $k$ ): Compute  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2) \leftarrow \text{PramGen}_a(k)$ . Choose  $x \xleftarrow{\text{R}} \mathbb{Z}_q^*$  and  $y \xleftarrow{\text{R}} \mathbb{Z}_q^*$ . Compute  $u_1 = g^x$  and  $u_2 = g^y$ . Output  $pk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, u_1, u_2)$  and  $sk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, u_1, u_2, x, y)$ .
- `Sign`( $sk, m$ ) where  $m \in \mathbb{Z}_q$ : Choose  $l \xleftarrow{\text{R}} \mathbb{Z}_q^*$  and compute  $\delta = g^{1/(x+m+yl)}$ . (If  $x + m + yl = 0$ , try different  $l$ ). Output  $\sigma = (\delta, l)$ .
- `Verify`( $pk, m, \sigma$ ): Check  $e(\sigma, u_1 g^m u_2^l) \stackrel{?}{=} e(g, g)$ . If the equality holds, output 1. Otherwise, output 0.
- `Transform`( $pk, \sigma$ ): Choose  $z \xleftarrow{\text{R}} \mathbb{Z}_q^*$  and compute  $\tilde{\delta} = \delta^z (= g^{z/(x+m+yl)})$ . Output  $\tilde{\sigma} = (\tilde{\delta}, l)$  and  $\tilde{sk} = z$ .
- `IVerify`[ $P(\tilde{sk}) \leftrightarrow V$ ]( $pk, \tilde{\sigma}, m$ ):
  - Both  $P$  and  $V$  compute  $v_1 = e(\tilde{\delta}, u_1 g^m u_2^l)$ , and  $v_2 = e(g, g)$ .
    1.  $P$  chooses  $s \xleftarrow{\text{R}} \mathbb{Z}_q^*$ , computes  $w = v_2^s$  and sends  $w$  to  $V$ .
    2.  $V$  chooses  $c \xleftarrow{\text{R}} \mathbb{Z}_q^*$  and sends it to  $P$ .
    3.  $P$  computes  $t = s + cz \pmod q$  and sends  $t$  to  $V$ .
    4.  $V$  checks  $v_1^t \stackrel{?}{=} w v_2^c$ . If the equality holds then output 1. Otherwise, output 0.

The above UDVSP system is structurally similar to the previous one presented in Section 4.1. However, a nice feature of this second construction is that it does not depend on random oracle due to the underlying signature scheme can be proven unforgeable against chosen message attack without employing the random oracle assumption [7].

Similarly to the previous construction, the `IVerify` protocol in the above UDVSP system can be viewed as a protocol for proving the knowledge of  $z$  satisfying the relation  $e(\tilde{\sigma}, u_1 g^m u_2^l) = e(g, g)^z$ .

## 5 Security Analysis

As mentioned in the previous section, the unforgeability of the underlying signature schemes of our two UDVSP systems were proven under the assumption

that the CDH and SDH problems are hard respectively [5, 7]. Therefore, we only analyze the security of the two UDVSP systems under impersonation attack.

The first theorem is concerned with the security of our UDVS system based on the BLS signature against impersonation under Type-1 attack.

**Theorem 1.** *The UDVSP system based on the BLS signature is secure against impersonation under Type-1 attack in the random oracle model assuming that the OMDL problem is hard in  $\mathbb{G}_1$ .*

*Proof.* Let  $A = (\hat{V}, \hat{P})$  be an impersonator that tries to break the UDVSP system based on the BLS signature under Type-1 attack. Let  $B$  be an OMDL attacker. Suppose that  $B$  is given  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2)$ . First,  $B$  queries its challenge oracle  $\mathcal{C}(\cdot)$  to obtain a challenge point  $h_0$ . Suppose that  $h_0 = g^{s_0}$  for some random  $s_0 \in \mathbb{Z}_q^*$ . Now,  $B$  chooses an arbitrary string  $m \in \{0, 1\}^*$ .  $B$  also chooses  $x \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$  and computes  $y = g^x$ .  $B$  then outputs  $pk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, H, y)$  as the signer's public key, where a random oracle  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$  can be constructed as follows.  $B$  sets  $H(m) = g^l \in \mathbb{G}_1$ . For  $m' \neq m$ ,  $B$  chooses  $l' \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$  and returns  $H(m') = g^{l'} \in \mathbb{G}_1$ . Finally,  $B$  computes  $\tilde{\sigma} = h_0^{lx}$  and gives this to  $A$  as a transformed signature.  $B$  proceeds to simulate the  $n$  times of execution of the IVerify protocol between  $\hat{V}$  and an honest designator  $P$  as follows. (Below,  $i \in \{1, \dots, n\}$ ).

- Make a query to  $\mathcal{C}(\cdot)$  and get the response  $h_i$ . Compute  $w_i = e(h_i^{lx}, g)$  and send this to  $\hat{V}$ . When  $\hat{V}$  sends  $c_i$ , make query  $h_i h_0^{c_i}$  to  $\mathcal{DL}_{q,g}(\cdot)$  to get the response  $t_i$  and send this back to  $\hat{V}$ .  $\hat{V}$  then checks  $e(H(m), g)^{t_i} \stackrel{?}{=} w_i e(\tilde{\sigma}, g)^{c_i}$ .

Notice that the distribution of the transformed signature  $\tilde{\sigma}$  constructed in the simulation above is the same as that in the real attack: Since  $h_0 = g^{s_0}$  for random  $s_0 \in \mathbb{Z}_q^*$ ,  $\tilde{\sigma} = h_0^{lx} = g^{lx s_0} = H(m)^{x s_0}$  for random  $l, x \in \mathbb{Z}_q^*$ .

We now show that the simulation of IVerify is correct. Firstly,  $w_i$  in the above simulation and the one sent by  $P$  in Step 1 of the real protocol are identically distributed: Since  $h_i = g^{s_i}$  for some random  $s_i \in \mathbb{Z}_q^*$ , we have

$$w_i = e(h_i^{lx}, g) = e(g^{s_i l}, g^x) = e(g^l, g^x)^{s_i} = e(H(m), y)^{s_i}.$$

Secondly, since  $t_i$  is the discrete-logarithm of  $h_i h_0^{c_i}$ , we have  $t_i = s_i + c_i s_0 \pmod q$  and hence

$$\begin{aligned} e(H(m), y)^{t_i} &= e(H(m), y)^{s_i + c_i s_0} = e(H(m), y)^{s_i} e(H(m), y)^{c_i s_0} \\ &= e(g^l, g^x)^{s_i} e(g^l, g^x)^{c_i s_0} = e(g^{s_i l}, g) e(g^{s_0 l}, g)^{c_i} \\ &= e(h_i^{lx}, g)^{s_i} e(h_0^{lx}, g)^{c_i} = w_i e(\tilde{\sigma}, g)^{c_i}. \end{aligned}$$

After performing the above simulation  $n$  times,  $B$  now attempts to extract  $s_0$ , the discrete-logarithm of  $h_0$ . Using this value,  $B$  can compute the discrete-logarithms of other points  $h_1, \dots, h_n$ . To do so,  $B$  runs  $\hat{P}$  to get  $w$  in Step

1 of IVerify, selects  $c \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$  and runs  $\hat{P}$  to obtain its response  $t$  and checks  $e(H(m), y)^t \stackrel{?}{=} we(\tilde{\sigma}, g)^c$ . If the equality holds,  $B$  runs  $\hat{P}$  again with the same state as before but with different challenge  $c' \in \mathbb{Z}_q^*$ , obtains its response  $t'$  and checks  $e(H(m), y)^{t'} \stackrel{?}{=} we(\tilde{\sigma}, g)^{c'}$ . If the equality holds,  $B$  computes  $\frac{t-t'}{c'-c} \bmod q$ . (Note that this part is due to the “reset lemma” formulated in [3]). We show that  $\frac{t-t'}{c'-c} \bmod q$  is the discrete-logarithm of  $h_0$ . Observe that

$$\begin{aligned} e(g^{lx(t-t')/(c-c')}, g) &= e(g^{lx(t-t')}, g)^{1/(c-c')} = (e(g^l, g^x)^t e(g^l, g^x)^{-t'})^{1/(c-c')} \\ &= (e(H(m), y)^t e(H(m), y)^{-t'})^{1/(c-c')} \\ &= (we(\tilde{\sigma}, g)^c (we(\tilde{\sigma}, g)^{c'})^{-1})^{1/(c-c')} \\ &= e(\tilde{\sigma}, g)^{(c-c')^{1/(c-c')}} = e(h_0^{lx}, g) = e(g^{lx s_0}, g). \end{aligned}$$

From the above equation, we obtain  $\frac{t-t'}{c'-c} = s_0 \bmod q$ . Since we have  $s_0$ , we can compute  $s_i = t_i - c_i(s_0 f_i)$  for  $i = 1, \dots, n$ . Finally  $B$  outputs  $s_0, s_1, \dots, s_n$ .

Now we prove the security of our first UDVS system against impersonation under Type-2 attack.

**Theorem 2.** *The UDVSP system based on the BLS signature is secure against impersonation under Type-2 attack in the random oracle model assuming that the CDH problem is hard in  $\mathbb{G}_1$ .*

*Proof.* We present a reduction from the unforgeability of the original BLS signature scheme to the security of our UDVSP system under Type-2 attack. The above theorem then follows since the BLS scheme is shown to be (existentially) unforgeable under chosen message attack in the random oracle model assuming that the CDH problem in  $\mathbb{G}_1$  is hard [5].

Let  $A$  be an impersonator that tries to break the UDVSP system based on the BLS signature under Type-2 attack. Let  $B$  be a forger that tries to break the BLS signature scheme under chosen message attack. Suppose that  $B$  is given a public key  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2, y, H)$ , where  $y = g^x$  and  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$  is a hash function modeled as a random oracle [2]. Firstly,  $B$  outputs  $pk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, H, y)$  as the signer’s public key.  $B$  then chooses an arbitrary string  $m \in \{0, 1\}^*$ .

$B$  now runs  $A$  to get  $\tilde{\sigma}'$ .  $B$  continues to run  $A$  to get  $w$  in Step 1 of IVerify. Upon receiving  $w$ ,  $B$  picks  $c \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$ , runs  $A$  to obtain its response  $t$  and checks  $e(H(m), y)^t \stackrel{?}{=} we(\tilde{\sigma}', g)^c$ . If the equality holds,  $B$  runs  $A$  again with the same state as before but with difference challenge  $c' \in \mathbb{Z}_q^*$ , obtains its response  $t'$  and checks  $e(H(m), y)^{t'} \stackrel{?}{=} we(\tilde{\sigma}', g)^{c'}$ . If the equality holds,  $B$  outputs  $\tilde{\sigma}'^{\frac{c-c'}{t-t'}}$  as a forgery. (Similarly to the proof of the previous theorem, this part is due to the “reset lemma” formulated in [3]).

It remains to show that  $\tilde{\sigma}'^{\frac{c-c'}{t-t'}}$  is a valid signature on  $m$ . From the above two equations, we get

$$\begin{aligned} e(H(m), y)^t / e(H(m), y)^{t'} &= we(\tilde{\sigma}', g)^c / we(\tilde{\sigma}', g)^{c'} \\ e(H(m), y)^{t-t'} &= e(\tilde{\sigma}', g)^{c-c'} \\ e(H(m), g^x)^{t-t'} &= e(\tilde{\sigma}', g)^{c-c'} \\ e(H(m)^x, g) &= e(\tilde{\sigma}', g)^{\frac{c-c'}{t-t'}} \\ e(H(m)^x, g) &= e(\tilde{\sigma}'^{\frac{c-c'}{t-t'}}, g). \end{aligned}$$

Thus we have  $\tilde{\sigma}'^{\frac{c-c'}{t-t'}} = H(m)^x$ , which is a valid signature on  $m$ .

The following two theorems are concerned with the security of our second UDVSP system against impersonation attack.

**Theorem 3.** *The UDVSP system based on the BB signature is secure against impersonation under Type-1 attack assuming that the OMDL problem is hard in  $\mathbb{G}_1$ .*

*Proof.* Let  $A = (\hat{P}, \hat{V})$  be an impersonator that tries to break the UDVSP protocol based on the BB signature scheme under Type-1 attack. Let  $B$  be an OMDL attacker. Suppose that  $B$  is given  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2)$ . First,  $B$  queries its challenge oracle  $\mathcal{C}(\cdot)$  to obtain a challenge point  $h_0$ . Suppose that  $h_0 = g^{s_0}$  for some random  $s_0 \in \mathbb{Z}_q^*$ .  $B$  then chooses an arbitrary string  $m \in \{0, 1\}^*$ .  $B$  also chooses  $x \xleftarrow{R} \mathbb{Z}_q^*$  and  $y \xleftarrow{R} \mathbb{Z}_q^*$  and computes  $u_1 = g^x$  and  $u_2 = g^y$ .  $B$  then outputs  $pk = (k, q, e, g, \mathbb{G}_1, \mathbb{G}_2, u_1, u_2)$  as the signer's public key. Finally,  $B$  computes  $\tilde{\sigma} = h_0^{\frac{1}{x+m+y}}$  and publishes this as a transformed signature.  $B$  proceeds to simulate the steps of  $\mathsf{IVerify}$  as follows. (Below,  $i \in \{1, \dots, n\}$ ).

- Make a query to  $\mathcal{C}(\cdot)$  and get the response  $h_i$ . Compute  $w_i = e(h_i, g)$  and send this to  $\hat{V}$ . When  $\hat{V}$  sends  $c_i$ , make query  $h_i h_0^{c_i}$  to  $\mathcal{DL}_{q,g}(\cdot)$  to get the response  $t_i$  and send this back to  $\hat{V}$ .  $\hat{V}$  then checks  $e(g, g)^{t_i} \stackrel{?}{=} w_i e(\tilde{\sigma}, u_1 g^m u_2^{c_i})$ .

Notice that the distribution of the transformed signature  $\tilde{\sigma}$  constructed in the simulation above is the same as that in the real attack as  $h_0 = g^{s_0}$  for random  $s_0 \in \mathbb{Z}_q^*$  and hence  $\tilde{\sigma} = h_0^{\frac{1}{x+m+y}} = g^{\frac{s_0}{x+m+y}}$  for random  $x, y, l \in \mathbb{Z}_q^*$ .

We now show that the simulation of  $\mathsf{IVerify}$  is correct. Firstly,  $w_i$  in the above simulation and the one sent by  $P$  in Step 1 of the real protocol are identically distributed: Since  $h_i = g^{s_i}$  for some random  $s_i \in \mathbb{Z}_q^*$ , we have  $w_i = e(g^{s_i}, g) = e(g, g)^{s_i}$ . Secondly, since  $t_i$  is the discrete-logarithm of  $h_i h_0^{c_i}$ , we have  $t_i = s_i + c_i s_0 \pmod{q}$ . Hence,

$$\begin{aligned} e(g, g)^{t_i} &= e(g, g)^{s_i + c_i s_0} = e(g, g)^{s_i} e(g, g)^{c_i s_0} = w_i e(g^{\frac{s_0}{x+m+y}}, g^{x+m+y})^{c_i} \\ &= w_i e(h_0^{\frac{1}{x+m+y}}, u_1 g^m u_2^{c_i})^{c_i} = w_i e(\tilde{\sigma}, u_1 g^m u_2^{c_i})^{c_i}. \end{aligned}$$

After performing the above simulation  $n$  times,  $B$  now attempts to extract  $s_0$ , the discrete-logarithm of  $h_0$ . Using this value,  $B$  can compute the discrete-logarithms of other points  $h_1, \dots, h_n$ . To do so,  $B$  runs  $\hat{P}$  to get  $w$  in Step 1 of **Verify**, selects  $c \xleftarrow{R} \mathbb{Z}_q^*$  and runs  $\hat{P}$  to obtain its response  $t$  and checks  $e(g, g)^t \stackrel{?}{=} we(\tilde{\sigma}, g)^c$ . If the equality holds,  $B$  runs  $\hat{P}$  again with the same state as before but with different challenge  $c' \in \mathbb{Z}_q^*$ , obtains its response  $t'$  and checks  $e(g, g)^{t'} \stackrel{?}{=} we(\tilde{\sigma}, g)^{c'}$ . If the equality holds,  $B$  computes  $\frac{t-t'}{c'-c} \bmod q$ . (Note that this part is due to the “reset lemma” formulated in [3]). We now show that  $\frac{t-t'}{c'-c} \bmod q$  is the discrete-logarithm of  $h_0$ . Observe that

$$\begin{aligned} e(g^{\frac{t-t'}{c'-c}}, g) &= e(g^{t-t'}, g)^{\frac{1}{(c'-c)}} = (e(g, g)^t e(g, g)^{-t'})^{\frac{1}{(c'-c)}} \\ &= \left( we(\tilde{\sigma}, u_1 g^m u_2^l)^c (we(\tilde{\sigma}, u_1 g^m u_2^l)^{c'})^{-1} \right)^{\frac{1}{(c'-c)}} \\ &= e(\tilde{\sigma}, u_1 g^m u_2^l)^{\frac{c-c'}{c'-c}} = e(h_0^{\frac{1}{x+m+yl}}, g^{x+m+yl}) = e(g^{s_0}, g). \end{aligned}$$

From the above equation, we obtain  $\frac{t-t'}{c'-c} = s_0 \bmod q$ . Since we have  $s_0$ , we can compute  $s_i = t_i - c_i s_0$  for  $i = 1, \dots, n$ . Finally  $B$  outputs  $s_0, s_1, \dots, s_n$ .

**Theorem 4.** *The UDVSP system based on the BB signature is secure against impersonation under Type-2 attack in the random oracle model assuming that the SDH problem is hard in  $\mathbb{G}_1$ .*

*Proof.* Similarly to the proof of Theorem 2, we present a reduction from the security of the original BB signature scheme to the security of our UDVSP system under Type-2 attack. The above theorem then follows since the BB scheme is shown to be unforgeable under chosen message attack (in the standard model) assuming that the SDH problem in group  $\mathbb{G}_1$  is hard [7].

Let  $A$  be an impersonator that tries to break the UDVSP system based on BB under Type-2 attack. Let  $B$  be a forger that tries to break the BB signature scheme under chosen message attack. Suppose that  $B$  is given a public key  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2, u_1, u_2)$ , where  $u_1 = g^x$  and  $u_2 = g^y$  for random  $x, y, \mathbb{Z}_q^*$ . Firstly,  $B$  outputs  $pk = (q, g, e, \mathbb{G}_1, \mathbb{G}_2, u_1, u_2)$  as the signer’s public key.  $B$  chooses an arbitrary string  $m \in \{0, 1\}^*$ .

$B$  now runs  $A$  to get  $\tilde{\sigma}' = (\tilde{\delta}', l)$ .  $B$  continues to run  $A$  to get  $w$  in Step 1 of **Verify**. Upon receiving  $w$ ,  $B$  picks  $c \xleftarrow{R} \mathbb{Z}_q^*$ , runs  $A$  to obtain its response  $t$  and checks  $e(g, g)^t \stackrel{?}{=} we(\tilde{\delta}', u_1 g^m u_2^l)^c$ . If the equality holds,  $B$  runs  $A$  again with the same state as before but with difference challenge  $c' \in \mathbb{Z}_q^*$ , obtains its response  $t'$  and checks  $e(g, g)^{t'} \stackrel{?}{=} we(\tilde{\delta}', u_1 g^m u_2^l)^{c'}$ . If the equality holds,  $B$  outputs  $\tilde{\sigma}^{\frac{t-t'}{c'-c}}$  as a forgery. (Note that this part is due to the “reset lemma” formulated in [3]).

We now show that  $\tilde{\sigma}'^{\frac{t-t'}{c-c'}}$  is a valid signature on the message  $m$ . From the above two equations, we get

$$\begin{aligned} e(g, g)^t / e(g, g)^{t'} &= we(\tilde{\sigma}', g)^c / we(\tilde{\sigma}', g)^{c'} \\ e(g, g)^{t-t'} &= e(\tilde{\delta}', u_1 g^m u_2^l)^{c-c'} \\ e(g, g)^{t-t'} &= e(\tilde{\delta}', u_1 g^m u_2^l)^{c-c'} \\ e(g, g) &= e(\tilde{\sigma}', g^{x+m+yl})^{\frac{c-c'}{t-t'}} \\ e(g, g) &= e(\tilde{\sigma}'^{\frac{c-c'}{t-t'}}(x+m+yl), g). \end{aligned}$$

Thus we have  $\tilde{\sigma}'^{\frac{t-t'}{c-c'}} = g^{\frac{1}{x+m+yl}}$ , which is a valid signature on  $m$ .

## 6 Concluding Remarks

In this paper, we proposed an alternate method to realize Universal Designated Verifier Signature (UDVS) [25], called Universal Designated Verifier Signature Proof (UDVSP). We constructed two *efficient* and *provably-secure* UDVSP systems based on the pairing-based signature schemes proposed in [5] and [7]. The important feature of our two constructions compared to previous UDVS schemes [25, 26, 28] is that the verifier is no longer required to generate his private/public key pair for verifying that the signature holder is in possession of a right signature from the original signer.

Additionally, we observe that *when a designator and a signer are the same entity*, the UDVSP system becomes an undeniable signature scheme with a confirmation protocol only. As an example, using the UDVSP system based on BLS signature, one can construct an undeniable signature scheme as follows.

- **SigKeyGen**( $k$ ): Compute  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2) \leftarrow \text{PramGen}_a(k)$ . Choose  $x \xleftarrow{\text{R}} \mathbb{Z}_q^*$  and compute  $y = g^x$ . Specify a hash function  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ . Output  $pk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, H, y)$  and  $sk = (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, H, x)$ .
- **Sign**( $sk, m$ ), where  $m \in \{0, 1\}^*$ : Choose  $z \xleftarrow{\text{R}} \mathbb{Z}_q^*$  and compute  $\tilde{\sigma} = H(m)^{xz}$ . Output  $\tilde{\sigma}$ .
- **Confirmation**[ $P(sk, z) \leftrightarrow V(pk, \tilde{\sigma}, m)$ ]:
  - Both  $P$  and  $V$  compute  $v_1 = e(\tilde{\sigma}, g)$ , and  $v_2 = e(H(m), y)$ .
    1.  $P$  chooses  $t \xleftarrow{\text{R}} \mathbb{Z}_q^*$ , computes  $w = v_2^t$  and sends  $w$  to  $V$ .
    2.  $V$  chooses  $c \xleftarrow{\text{R}} \mathbb{Z}_q^*$  and sends it to  $P$ .
    3.  $P$  computes  $t = s + cz \pmod q$  and sends  $t$  to  $V$ .
    4.  $V$  checks  $v_2^t \stackrel{?}{=} wv_1^c$ . If the equality holds then output 1. Otherwise, output 0.

Notice that in the above scheme, anyone cannot verify the validity of a signature without the help of the signer, which is essentially a main requirement of undeniable signature.

Designing a disavowal protocol for the above scheme and its formal analysis under the new security model for undeniable signature proposed in [22] are our future work.

## Acknowledgement

The authors are grateful to the anonymous referees of Asiacrypt 2005 for their helpful comments.

## References

1. M. Bellare and O. Goldreich, *On Defining Proof of Knowledge*, In Crypto '92, LNCS 740, pp. 390–420, Springer-Verlag, 1993.
2. M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In ACM-CCS, pp. 62–73, 1993.
3. M. Bellare and A. Palacio, *GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks*, In Crypto '02, LNCS 2442, pp. 162–177, Springer-Verlag, 2002.
4. D. Boneh and M. Franklin, *Identity-Based Encryption from the Weil Pairing*, In Crypto '01, LNCS 2139, pp. 213–229, Springer-Verlag, 2001.
5. D. Boneh, B. Lynn and H. Shacham, *Short Signatures from the Weil Pairing*, In Asiacrypt '01, LNCS 2248, pp. 566–582, Springer-Verlag, 2001.
6. D. Boneh, C. Gentry, B. Lynn and H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, In Eurocrypt '03, LNCS 2656, pp. 416–432, Springer-Verlag, 2003.
7. D. Boneh and X. Boyen, *Short Signatures Without Random Oracles*, In Eurocrypt '04, LNCS 3027, pp. 56–73, Springer-Verlag, 2004.
8. G. Brassard, D. Chaum and C. Crpeau, *Minimum Disclosure Proof of Knowledge*, In Journal of Computer and System Sciences, 37 (2), pp. 156–189, 1988.
9. D. Chaum and H. Antwerpen, *Undeniable Signatures*, In Crypto '89, LNCS 435, pp. 212–216, Springer-Verlag, 1990.
10. J. Camenisch and A. Lysyanskaya, *An Efficient System for Non-Transferable Anonymous Credentials with Anonymity Revocation*, In Eurocrypt '01, LNCS 2045, pp. 93–118, Springer-Verlag, 2001.
11. J. Camenisch and A. Lysyanskaya, *Signature Schemes and Anonymous Credentials from Bilinear Maps*, In Crypto '04, LNCS 3152, pp. 56–72, Springer-Verlag, 2004.
12. Y. Desmedt, C. Gourtier and S. Bengio, *Special Uses and Abuses of the Fiat-Shamir Passport Protocol*, In Crypto '87, LNCS 293, pp. 21–39, Springer-Verlag, 1988.
13. U. Feige and A. Shamir, *Witness Indistinguishability and Witness Hiding Protocols*, In 22nd Symposium on the Theory of Computing (STOC), pp. 416–426, ACM, 1990.
14. S. Goldwasser, S. Micali and R. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attack*, In SIAM Journal on Computing, 17 (2), pp. 281–308, 1988.
15. S. Goldwasser, S. Micali and C. Rackoff, *The Knowledge Complexity of Interactive Proof System*, In SIAM Journal on Computing, 18 (1), pp. 186–208, 1989.
16. M. Jakobsson, *Blackmailing Using Undeniable Signatures*, In Eurocrypt '94, LNCS 950, pp. 425–427, Springer-Verlag, 1994.
17. M. Jakobsson, K. Sako and R. Impagliazzo, *Designated Verifier Proofs and Their Applications*, In Eurocrypt '96, LNCS 1070, pp. 143–154, Springer-Verlag, 1996.
18. A. Joux: *The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems*, Algorithmic Number Theory Symposium (ANTS-V) '02, LNCS 2369, pp. 20–32, Springer-Verlag, 2002.



19. H. Krawczyk and T. Rabin, *Chameleon Hashing and Signatures*, Network and Distributed System Security Symposium (NDSS) '00, pp. 143 – 154, The Internet Society, 2000.
20. A. Lysyanskaya, R. Rivest, A. Sahai and S. Wolf *Pseudonym Systems*, In Selected Areas in Cryptography (SAC) '99, LNCS 1758, pp. 184–199, Springer-Verlag, 1999.
21. M. Naor, *Deniable Ring Authentication*, In Crypto '02, LNCS 2442, pp. 481–198, Springer-Verlag, 2002.
22. W. Ogata, K. Kurosawa and S. Heng, *The Security of the FDH Variant of Chaum's Undeniable Signature Scheme*, In Public Key Cryptography (PKC) '05, LNCS 3386, pp. 328–345, Springer-Verlag, 2005.
23. C. P. Schnorr, *Efficient Identifications and Signatures for Smart Cards*, In Crypto '89, LNCS 435, pp. 239–251, Springer-Verlag, 1990.
24. R. Rivest, A. Shamir and Y. Tauman, *How to Leak a Secret*, In Asiacrypt '01, LNCS 2248, pp. 552 – 565, Springer-Verlag, 2001.
25. R. Steinfeld, L. Bull, H. Wang and J. Pieprzyk, *Universal Designated-Verifier Signatures*, In Asiacrypt '03, LNCS 2894, pp. 523–542, Springer-Verlag, 2003.
26. R. Steinfeld, H. Wang and J. Pieprzyk, *Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures*, In Public Key Cryptography (PKC) '04, LNCS 2947, pp. 86–100, Springer-Verlag, 2004.
27. W. Susilo and Y. Mu, *Deniable Ring Authentication Revisited*, In Applied Cryptography and Network Security (ACNS) '04, LNCS 3089, pp. 149–163, Springer-Verlag, 2004.
28. R. Zhang, J. Furukawa and H. Imai, *Short signature and Universal Designated Verifier Signature without Random Oracles*, In Applied Cryptography and Network Security (ACNS) '05, LNCS 3531, pp. 483–498, Springer-Verlag, 2005.

## A Definitions of CDH and SDH

**Definition 6 (CDH).** Let  $A$  be a polynomial-time attacker. Consider the following experiment  $\mathbf{Exp}^{cdh}(k)$ :

- Experiment:  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2) \leftarrow \mathbf{ParamGen}_a(k)$ ;  $(a, b) \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$ ;  $\kappa \leftarrow A(g^a, g^b)$
- Output: If  $\kappa = g^{ab}$  then return 1. Otherwise, return 0.

We define  $A$ 's advantage as  $\mathbf{Adv}_A^{cdh}(k) = \Pr[\mathbf{Exp}^{cdh}(k)] - \frac{1}{q}$ . We say that CDH problem is hard if  $\mathbf{Adv}_A^{cdh}(k)$  is negligible in  $k$ .

It is believed that the above CDH problem in group  $\mathbb{G}_1$  is hard (computationally intractable). On the contrary, the Decisional Diffie-Hellman (DDH) problem in this group can be solved in polynomial time with the help of the bilinear pairing. We note that the security of Boneh, Lynn and Shacham's [5] short signature scheme is relative to CDH.

We now review the definition of the Strong Diffie-Hellman (SDH) problem in group  $\mathbb{G}_1$  as follows.

**Definition 7 (SDH).** Let  $A$  be a polynomial-time attacker. Consider the following experiment  $\mathbf{Exp}^{sdh}(k)$  (Below,  $n$  is polynomial in  $k$ ):

- Experiment:  $(q, g, e, \mathbb{G}_1, \mathbb{G}_2) \leftarrow \text{PrmGen}_a(k)$ ;  $x \xleftarrow{\text{R}} \mathbb{Z}_q^*$ :  
 $\kappa \leftarrow A(g, g^x, g^{x^2}, \dots, g^{x^n})$
- Output: If  $\kappa = (c, g^{1/(x+c)})$  where  $c \in \mathbb{Z}_q$  then return 1. Otherwise, return 0.

We define  $A$ 's advantage as  $\mathbf{Adv}_A^{sdh}(k) = \Pr[\mathbf{Exp}^{sdh}(k)] = 1$ . We say that SDH problem is hard if  $\mathbf{Adv}_A^{sdh}(k)$  is negligible in  $k$ .

We note that the security of Boneh and Boyen's [7] signature scheme is relative to SDH.