SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.820

SPECIAL ISSUE PAPER

Secure universal designated verifier identity-based signcryption †

Changlu Lin[1]*, Fei Tang[2], Pinhui Ke[1], Lein Harn[3] and Shengyuan Zhang[1]

[1] Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350007, China
[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] Department of Computer Science Electrical Engineering, University of Missouri-Kansas City, MO 64110, U.S.A.

ABSTRACT

In 2003, Steinfeld *et al*. introduced the notion of universal designated verifier signature (UDVS), which allows a signature holder, who receives a signature from the signer, to convince a designated verifier whether he is possession of a signer's signature; at the same time, the verifier cannot transfer such conviction to anyone else. These signatures devote to protect the receiver's privacy, that is, the receiver may want to prove to any designated verifier who he is in possession of such signature signed by the known signer but reluctant to disclose it. Moreover, the receiver also does not want the verifier to be able to convince anyone that he is in possession of such signature. In the existing UDVS schemes, a secure channel is required between the signer and the signature holder to transfer the signature. This paper, for the first time, proposes the notion of *universal designated verifier signcryption* without this secure channel by combining the notions of UDVS and signcryption. We give the formal definitions and a concrete construction of universal designated verifier identity-based signcryption scheme. We also give the formal security proofs for our scheme under the random oracle model. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS
universal designated verifier; signcryption scheme; identity-based cryptography; random oracle model

***Correspondence**
Changlu Lin, Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350007, China.
E-mail: cllin@fjnu.edu.cn

1. INTRODUCTION

Jakobsson *et al*. [1] proposed the concept of designated verifier signature (say, DVS). In a DVS scheme, an entity, called as the designated verifier, can produce signatures, which are indistinguishable from those generated by the signer, whereas any third party cannot distinguish a signature generated by the signer from a signature generated by the designated verifier. Then, a signature produced by the signer can only convince the designated verifier chosen by the signer. Because this is a good property, DVS scheme is very useful in many scenarios, such as call for tenders and digital vote, in which, the signer's privacy is a concern, and there are a number of designs [1–3] in the existing research works.

Steinfeld *et al*. [2], motivated by the privacy on disseminating signed digital certificates, first proposed the notion of *universal designated verifier signature* (say, UDVS) in 2003. There are three parties, such as, a signer, a signature holder, and a (multi-)designated verifier in the UDVS scheme. The signer signs a message with his private key and then sends this signature to the receiver, who is called as the signature holder. To claim that the signer has signed the message for some entity, the signature holder is required to send the signature to this entity. However, this entity also can convince anyone else about the signer's endorsement on the message. A more feasible way is that the signature holder converts the signer's signature into a UDVS signature, in which, a verifier is designated, such that this designated verifier can be convinced of the signer's endorsement on the message only. Meanwhile, any other third parties will not assure it as the designated verifier can use his/her private key to generate one valid UDVS signature, which is indistinguishable from the signature produced by the signature holder. On the

† This paper is the extended version of the previous publication in proceedings of the sixth international conference on Network and System Security, LNCS 7645, 2012.

other hand, the designated verifier in UDVS scheme can be convinced of the signer has signed the message, as he/she did not generate that UDVS signature. A DVS scheme is a special case of UDVS scheme if the signature holder and the signature signer act as the same entity.

In all existing UDVS schemes, to prevent signature exposure, there is a secure channel between the signer and the receiver to deliver the signature. In this paper, we introduce the concept of *universal designated verifier signcryption* (say, UDVSC), for the first time, combining the notions of UDVS scheme and signcryption scheme. This new notion not only keeps all desirable properties of UDVS but also eliminates the need of secure channel for signature delivery. As shown in [4], a secure signcryption scheme provides both confidentiality and authentication in a more efficient way than signature add encryption, that is, *Cost (Signcryption)* << *Cost (Signature)* + *Cost (Encryption)*.

ID-based cryptosystem, given first by Shamir [5], which simplifies the key management problem in public key cryptosystem with certificate authority (CA). Actually, in an ID-based cryptosystem, each user's public key is derived from his/her recognized identity information, for example, email address, telephone number, and so on, and then the user public key would be directly verified without certificate. In this paper, we adopt Chen *et al*.'s [6] signcryption scheme to construct a concrete ID-based UDVSC scheme.

Related works. There are some approaches to construct the UDVS scheme in the existing research works. We do some comments as follows.

(1) The notion of UDVS was first introduced by Steinfeld *et al*. [2] in the Asiacrypt'03. Steinfeld *et al*. [7] then extended the standard Schnorr and RSA signature schemes to UDVS schemes. Ng *et al*. [8] extended UDVS scheme to allow a signature holder to designate the signature to multi-verifiers (say, UDMVS). Shahandashti *et al*. [9] described a generic construction for UDVS scheme from a large class of signature schemes.

(2) Zhang *et al*. [10] designed a short UDVS scheme based on Boneh–Boyen's signature scheme [11] in the standard model. And then, Laguillaumie *et al*. [12], Huang *et al*. [13], and Vergnaud [14] also proposed UDVS schemes in the standard model, and Huang *et al*. [13] formalized the security models of UDVS scheme. Zhang *et al*. [15] first introduced the concept of UDVS schemes in identity-based settings (say, ID-based UDVS scheme) and provided the formal security model. Cao *et al*. [16] then proposed an ID-based UDVS scheme based on Waters' signature scheme [17] in the standard model. Seo *et al*. [18] proposed an ID-UDVS scheme in multi-verifiers settings in 2008.

(3) In 2005, Baek *et al*. [19] proposed two universal designated verifier signature proof (say, UDVSP) schemes based on Boneh–Lynn–Shacham's signature scheme [20] and Boneh–Boyen's signature scheme [11], respectively. In their schemes, they adopted an interactive proof protocol and eliminated the register process of designated verifier's secret-public key pair. Similarly, Chen *et al*. [21] also designed an identity-based UDVSP system with the similar interactive proof protocol. However, Li *et al*. [22] attacked the UDVSP systems proposed in [19,21] and showed that their schemes do not satisfy the property of non-transferability. To be more specific, a malicious designated verifier in UDVSP schemes [19,21] can transfer the interactive proof technology into a non-interactive signature scheme in which it can convince anyone that the signer has endorsed the message.

In addition, some UDVS schemes, with additional properties, have been proposed, such as, the restricted UDVS [23,24] scheme, the UDVS scheme without delegatability [25], the UDVS scheme with multi-signer [26], and the universal designated verifier ring signature [27] scheme.

Our contributions. We combine the concepts of UDVS and signcryption schemes, and introduce the notion of UDVSC, for the first time, in this paper. We present the formal definitions and a concrete construction of UDVSC in the identity-based setting (say, ID-UDVSC). We also do describe the formal security proofs for our scheme under the random oracle model. Our scheme shares the following properties: (i) The signer signcrypts some message and then sends this ciphertext to a designated receiver *without* a secure channel. That is, the signcrypted message can be transmitted via a public channel in our scheme, while there is a secure channel to transfer ciphertext in previous UDVS schemes. (ii) When the designated receiver receives the ciphertext, he/she can *efficiently* recover the original signature and transforms it into a transformational signature, and then delivers the new signature to a designated verifier. (iii) Finally, the designated verifier would *easily* verify the validity of the transformational signature; however, he/she cannot convince anyone else that the signer has endorsed the message. This saves the good property in the UVDS scheme.

Organization of this paper. In the next section, we review some preliminaries. Section 3 defines the formal definitions and security models of the new ID-UDVSC scheme. Section 4 describes the concrete construction of our ID-UDVSC scheme. In Section 5, we analyze the security of our scheme in the random oracle model. We conclude this paper in Section 6.

2. PRELIMINARIES

Notations. Some notations will be used in our paper: "$a \leftarrow \mathsf{A}(x)$", the short left arrow denotes an algorithm A with input x and then output a; "$w := v$" denotes the assignment of a value v to w; for simplicity, id_s, id_{dr}, id_{dv}, and PKG denote the signer, designated receiver, the designated verifier, and the trusty private key generator, respectively; sk_i denotes the user i's secret key, where

$i \in \{\mathsf{id}_s, \mathsf{id}_{dr}, \mathsf{id}_{dv}\}$, that is, i means some identity of the entities.

Bilinear Mapping. Let G_1, G_2, and G_T be two cyclic multiplicative groups with the same prime order p, where p is a secure and large prime such that it is computationally infeasible to solve the **discrete logarithm** problem in these three groups; and let g_1 and g_2 be an arbitrary generator of G_1 and G_2, respectively. We claim that a mapping $\hat{e} : G_1 \times G_2 \to G_T$ is an admissible bilinear mapping if it satisfies three properties as follows:

- *Bilinearity*: $\hat{e}\left(g_1^a, g_2^b\right) = \hat{e}(g_1, g_2)^{ab}$, for all $a, b \in Z_p$;
- *Non-degeneracy*: there exists $g_1^c \in G_1, g_2^d \in G_2$, for $c, d \in Z_p$, such that $\hat{e}\left(g_1^c, g_2^d\right) \neq 1_{G_T}$, where 1_{G_T} denotes G_T's identical element; and
- *Computability*: there exists an efficient algorithm to compute $\hat{e}\left(g_1^a, g_2^b\right)$ for all $a, b \in Z_p$.

We also assume there is an isomorphism $\psi : G_2 \to G_1$ such that $\psi(g_2) = g_1$. In a special case, we usually assume $G_2 = G_1$ and $g_2 = g_1 = g$ in some applications for simplicity. The readers are referred to [28] for more on the definition and design of bilinear mapping.

Bilinear Diffie–Hellman problem (BDH). We assume that g is chosen randomly from G_1, and $a, b, c \in Z_p^*$, then for g^a, g^b, g^c, the BDH problem is how to compute the value $\hat{e}(g, g)^{abc} \in G_T$.

It is generally accepted that the BDH problem is computationally infeasible with a suitable bilinear mapping.

3. FORMAL DEFINITIONS AND SECURITY MODELS

3.1. Formal definitions

Our ID-UDVSC scheme follows seven algorithms: the system setup, the users' secret key extracting, the signcrypting, the unsigncrypting, the transforming, the transforming by the designated verifier, and the designated verifying. They are denoted by Setup, Extract, Signcrypt, Unsigncrypt, Transform, $\overline{\text{Transform}}$, and D-Verify, respectively. The detailed descriptions of them are as follows.

- Setup: It is a probabilistic algorithm, and takes a security parameter k as input, while it outputs the master key msk and public parameters $params$ for the system. It can be indicated by $(msk, params) \leftarrow \mathsf{Setup}(k)$.
- Extract: It is a probabilistic algorithm, and takes id as input, while it outputs user id's secret key sk_{id}. It can be indicated by $sk_{\mathsf{id}} \leftarrow \mathsf{Extract}_{msk}(\mathsf{id})$.
- Signcrypt: It is a probabilistic algorithm, and takes $(m, sk_s, \mathsf{id}_{dr})$ as inputs, while it outputs a ciphertext δ. It can be indicated by $\delta \leftarrow \mathsf{Sign}_{sk_s, \mathsf{id}_{dr}}(m)$.
- Unsigncrypt: It is a deterministic algorithm, and takes (δ, sk_{dr}) as inputs, while it outputs the signature σ that is sealed in δ if it is valid; otherwise, it outputs *Rej*. It can be indicated by "σ/*Rej*" $\leftarrow \mathsf{Unsigncrypt}_{sk_{dr}}(\delta)$,
- Transform: It is a probabilistic algorithm, and takes $(\sigma, \mathsf{id}_{dv})$ as inputs, while it outputs a transformational signature $\tilde{\sigma}$. It can be indicated by $\tilde{\sigma} \leftarrow \mathsf{Transform}_{\mathsf{id}_{dv}}(\sigma)$.
- $\overline{\text{Transform}}$: It is a probabilistic algorithm, and takes $(m', \mathsf{id}_s, sk_{dv})$ as inputs, while it outputs a transformational signature $\tilde{\sigma}'$ on behalf of the designated verifier. It can be indicated by $\tilde{\sigma}' \leftarrow \overline{\mathsf{Transform}}_{sk_{dv}, \mathsf{id}_s}(m')$.
- D-Verify: It is a deterministic algorithm, and takes $(\tilde{\sigma}, sk_{dv}, \mathsf{id}_s)$ as inputs, while it outputs *Acc* if it is a valid transformational signature; otherwise, it outputs *Rej*. It can be indicated by "*Acc*/*Rej*" $\leftarrow \mathsf{D\text{-}Verify}_{sk_{dv}}(\tilde{\sigma})$.

3.2. Security models

Confidentiality. The confidentiality of ID-UDVSC scheme is necessary to ensure that *only* the designated receiver can obtain the signature. We describe the following experiment $\mathsf{Exp}_{\mathcal{A}}^{ind\text{–}iudvsc\text{–}cca}(k)$, which is played between an IND-IDUDVSC-CCA adversary, denoted as $\mathcal{A}$, and a challenger, denoted as $\mathcal{C}$.

- **Setup**: $\mathcal{C}$ runs this algorithm to obtain the master key msk and public parameters $params$, and distributes $params$ to $\mathcal{A}$.
- **Queries***(phase 1)*: $\mathcal{A}$ does some oracle queries to $\mathcal{C}$ and $\mathcal{C}$ should answer them with some information.
  - Extract queries: $\mathcal{A}$ gives id as input for this oracle, and $\mathcal{C}$ returns sk_{id} to $\mathcal{A}$ as the answer.
  - Signcrypt queries: $\mathcal{A}$ takes $(m, \mathsf{id}_s, \mathsf{id}_{dr})$ as inputs to this oracle, and $\mathcal{C}$ returns a ciphertext δ to $\mathcal{A}$ as the answer.
  - Unsigncrypt queries: $\mathcal{A}$ takes $(\delta, \mathsf{id}_{dr})$ as inputs to this oracle, and $\mathcal{C}$ returns σ if it is valid, else *Rej*, to $\mathcal{A}$ as the answer.
  - Transform queries: $\mathcal{A}$ takes $(\sigma, \mathsf{id}_{dv})$ as inputs to this oracle, and $\mathcal{C}$ answers $\mathcal{A}$ with the transformational signature $\tilde{\sigma}$.
  - $\overline{\text{Transform}}$ queries: $\mathcal{A}$ takes $(m', \mathsf{id}_s, \mathsf{id}_{dv})$ as inputs to this oracle, and $\mathcal{C}$ answers $\mathcal{A}$ with a transformational signature $\tilde{\sigma}'$.
  - D-Verify queries: $\mathcal{A}$ takes $(\tilde{\sigma}, \mathsf{id}_{dv})$ as inputs to this oracle, and $\mathcal{C}$ returns *Acc* if it is valid; otherwise, return *Rej*.
- **Challenge**: After finishing the execution of phase 1, the adversary $\mathcal{A}$ outputs two messages $\{m_0, m_1\}$ together with two identities $\{\mathsf{id}_s^*, \mathsf{id}_{dr}^*\}$, which are challenged messages. $\mathcal{C}$ randomly chooses a bit $b \in \{0, 1\}$ and then runs the signcrypted

algorithm to obtain the ciphertext δ^*. Finally, $\mathcal{C}$ sends it to $\mathcal{A}$.

- **Queries***(phase 2)*: After receiving δ^*, the adversary $\mathcal{A}$ can make more queries as in phase 1 but

  (1) id_{dr}^* is never taken as input during Extract queries.
  (2) $(\delta^*, \mathsf{id}_{dr}^*)$ is never taken as input during Unsigncrypt queries.

- **Guess**: At last, the adversary $\mathcal{A}$ outputs his guess random bit b' from $\{0,1\}$. We claim that $\mathcal{A}$ wins this experiment if $b' = b$. We say that the advantage of $\mathcal{A}$ in this experiment is $\mathbf{Adv}_{\mathcal{A}}^{\text{ind–idudvsc–cca}}(k) = \left|\mathbf{Pr}\left[b' = b\right] - \frac{1}{2}\right|$.

Definition 1. *We claim that an ID-UDVSC scheme is IND-IDUDVSC-CCA secure if there is no polynomial-bounded adversary to win the experiment* $Exp_{\mathcal{A}}^{ind–idudvsc–cca}(k)$ *with a non-negligible probability.*

Unforgeability. We improve the Huang *et al.*'s [13] security definition of UDVS scheme, *existential unforgeability of the transformational signature against adaptively chosen message attacks*, in our ID-UDVSC scheme. We consider the following experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{eu–idudvsc–cma}}(k)$ which is played between an EU-IDUDVSC-CMA adversary, denoted as $\mathcal{A}$, and a challenger, denoted as $\mathcal{C}$.

- **Setup:** It is the same as in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{ind–idudvsc–cca}}(k)$.
- **Queries**: it is same as in the phase 1 of Queries in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{ind–idudvsc–cca}}(k)$.
- **Output**: We claim that $\mathcal{A}$ wins this experiment if $\mathcal{A}$ obtains the output as a transformational signature $\tilde{\sigma^*}$ together with identities $\mathsf{id}_s^*, \mathsf{id}_{dv}^*$ such that

  (1) $Acc \leftarrow \mathsf{D\text{-}Verify}_{sk_{dv}^*}\left(\tilde{\sigma^*}\right)$.
  (2) id_s^* and id_{dv}^* are never taken as inputs during the Extract queries.
  (3) $\left(m^*, \mathsf{id}_s^*\right)$ is never taken as input during the Signcrypt queries, where m^* is the message corresponding to the forgery.
  (4) $\left(m^*, \mathsf{id}_s^*, \mathsf{id}_{dv}^*\right)$ is never taken as input during the $\overline{\mathsf{Transform}}$ queries.

Definition 2. *We claim that an ID-UDVSC scheme is EU-IDUDVSC-CMA secure if there exist no polynomial-bounded adversary to win the experiment* $Exp_{\mathcal{A}}^{eu–idudvsc–cma}(k)$ *with a non-negligible probability.*

Non-transferability. The purpose of non-transferability is to protect the signature receiver's privacy. It means that the designated verifier cannot convince any third party to believe that the original signer has signed on the message m. We also improve Huang *et al.*'s [13] security model of UDVS scheme, *non-transferability against adaptively chosen message attack*, into our ID-UDVSC scheme. We consider the experiment $\mathsf{Exp}_{\mathcal{D}}^{\text{nt–idudvsc–cma}}(k)$, as follows, which is played between a challenger, denoted as $\mathcal{C}$, and a distinguisher, denoted as $\mathcal{D}$.

- **Setup**: It is the same as in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{eu–idudvsc–cma}}(k)$ swapping $\mathcal{A}$ with the distinguisher $\mathcal{D}$.
- **Queries***(phase 1)*: It is the same as in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{eu–idudvsc–cma}}(k)$ swapping $\mathcal{A}$ with the distinguisher $\mathcal{D}$.
- **Challenge**: After finishing the execution of phase 1, $\mathcal{D}$ submits $\left(m^*, \mathsf{id}_s^*, \mathsf{id}_{dv}^*\right)$ to $\mathcal{C}$ as the challenge. Then, $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$ and

  (1) If $b = 1$, then the challenger $\mathcal{C}$ runs the Transform algorithm to obtain a $\tilde{\sigma}^*$, and then sends it to $\mathcal{D}$.
  (2) If $b = 0$, then the challenger $\mathcal{C}$ runs the $\overline{\mathsf{Transform}}$ algorithm to obtain a $\tilde{\sigma}^{*'}$, and then sends it to $\mathcal{D}$.

- **Queries***(phase 2)*: After $\mathcal{D}$ receives the σ^* or $\tilde{\sigma}^{*'}$, $\mathcal{D}$ can do some other queries as in phase 1.
- **Guess**: At last, $\mathcal{D}$ outputs the guess bit b' from $\{0,1\}$. We say that $\mathcal{D}$ wins the this experiment if $b' = b$. We define the advantage of $\mathcal{D}$ is $\mathbf{Adv}_{\mathcal{D}}^{\text{nt–idudvsc–cma}}(k) = |\mathbf{Pr}[b' = b] - \frac{1}{2}|$.

Definition 3. *We claim that an ID-UDVSC scheme is NT-IDUDVSC-CMA secure if there is no polynomial-bounded distinguisher to win the experiment* $Exp_{\mathcal{D}}^{nt–idudvsc–cma}(k)$ *with a non-negligible probability.*

Remark: In this experiment, $\mathcal{D}$ wants to obtain some information via making queries to the previous oracles, and then distinguish the transformational signature $\tilde{\sigma}^{*'}$ (generated by the designated verifier) from $\tilde{\sigma}^*$ (generated by the designated receiver). Our definition requires that nobody can know the difference between those two kinds of signatures.

4. PROPOSED ID-UDVSC SCHEME

We construct our ID-UDVSC scheme using the bilinear pairing together with the seven algorithms discussed later in this section.

- Setup: Run this algorithm to obtain the system parameters and the master key:

  (1) $G_1, G_T, \hat{e}, p, g$ have been described in Section 2.
  (2) It chooses random x from Z_p^* as the master key and computes the master public key as $y = g^x \in G_1$.

(3) $H_0 : \{0,1\}^{k_0} \to G_1, H_1 : \{0,1\}^{k_1+n} \to Z_p^*$ and $H_2 : G_T \to \{0,1\}^{k_0+k_1+n}$ are three collision-resistant hash functions, where k_0, k_1, n denote the sizes of users' identity, element of G_1, message m, respectively.
(4) Finally, it publishes the following system parameters $params :=< G_1, G_T, \hat{e}, p, g, y, H_0, H_1, H_2 >$, and keeps $msk := x$ secretly as the master key.

- Extract: PKG computes $sk_{\mathsf{id}} = H_0(\mathsf{id})^x \in G_1$ as user's secret key, and distributes the secret key to the user under a secure channel.
- Signcrypt: Let $m \in \{0,1\}^n$ be signcrypted by id_s who chooses random $r \in Z_p$ and computes as follows:

$$R = H_0(\mathsf{id}_s)^r, \qquad h = H_1(R\|m),$$
$$S = sk_s^{(r+h)}, \qquad T = \hat{e}(sk_s^r, H_0(\mathsf{id}_{dr})),$$
$$\alpha = H_2(T) \oplus (S\|\mathsf{id}_s\|m).$$

The ciphertext is $\delta := (R, \alpha)$.
- Unsigncrypt: After receiving δ, id_{dr} recovers the original signature and verifies its validity as follows:

$$T = \hat{e}(R, sk_{dr}), \quad (S\|\mathsf{id}_s\|m) = \alpha \oplus H_2(T),$$
$$h = H_1(R\|m), \quad \hat{e}(S, g) \stackrel{?}{=} \hat{e}(y, R \cdot H_0(\mathsf{id}_s)^h).$$

it outputs $\sigma := (m, R, S, \mathsf{id}_s)$ if the verification equation holds; otherwise, it outputs *Rej*.
- Transform: If id_{dr} wants to prove the signature σ to id_{dv}, he computes $V = \hat{e}(S, H_0(\mathsf{id}_{dv}))$. And, the transformational signature is $\tilde{\sigma} := (m, R, V, \mathsf{id}_s)$.
- $\overline{\text{Transform}}$: id_{dv} also can generate a transformational signature $\tilde{\sigma}' := (m', R', V', \mathsf{id}_s)$ for an arbitrary message m' from $\{0,1\}^n$, and then chooses random $R' \in G_1$ and computes

$$h = H_1(R'\|m'), \quad V' = \hat{e}(sk_{dv}, R' \cdot H_0(\mathsf{id}_s)^h).$$

- D-Verify: id_{dv} can verify $\tilde{\sigma}$'s validity via checking $V \stackrel{?}{=} \hat{e}(sk_{dv}, R \cdot H_0(\mathsf{id}_s)^h)$, where $h = H_1(R\|m)$, holds or not, it outputs *Acc* if it holds; otherwise, it outputs *Rej*.

5. SECURITY RESULTS

We present the security results of our ID-UDVSC scheme with the following three theorems and give the formal proofs for them in this section.

Theorem 1 (IND-IDUDVSC-CCA). *The proposed ID-UDVSC scheme is indistinguishable against adaptively chosen ciphertext attack if the BDH problem is hard to resolve.*

Proof. Assume there exists an IND-IDUDVSC-CCA adversary, $\mathcal{A}$, wins the defined experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{ind–idudvsc–cca}}(k)$ with a non-negligible probability ϵ, we will design a challenger, $\mathcal{C}$, who takes the adversary, $\mathcal{A}$, as the subroutine to solve the BDH problem with a non-negligible probability. The challenger, $\mathcal{C}$, is appointed with an instance $(G_1, G_T, \hat{e}, p, g, g^a, g^b, g^c)$ from the BDH problem, and will try to solve the given instance using $\mathcal{A}$ as a subroutine, that is, try to compute the value $\hat{e}(g,g)^{abc} \in G_T$.

At first, the challenger $\mathcal{C}$ sets $y := g^c$ as the master public key, and randomly chooses id_A as an identity. $\mathcal{C}$ maintains six lists L_0, L_1, L_2, L_u, L_t, and L_t' corresponding to the H_0, H_1, H_2, Unsigncrypt, Transform, and $\overline{\text{Transform}}$ oracles, respectively. Then, $\mathcal{C}$ plays the following experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{ind–idudvsc–cca}}(k)$ with $\mathcal{A}$. Furthermore, we assume that id_A queries q_0 and q_2 times to H_0 and H_2 oracles, respectively.

- **Setup:** $\mathcal{C}$ sends $params :=< G_1, G_T, \hat{e}, p, g, y, H_0, H_1, H_2 >$ to $\mathcal{A}$, where $(G_1, G_T, \hat{e}, p, g)$ is an instance of the BDH problem, $y := g^c$. The hash functions $H_i(\cdot), i = 0, 1, 2$, controlled by $\mathcal{C}$, are regarded as random oracles.
- **Queries***(phase 1)*: $\mathcal{A}$ do some queries to $\mathcal{C}$, and then $\mathcal{C}$ will answer them as follows.

(1) H_0 *oracle*: id_i. At the beginning, $\mathcal{C}$ chooses random $j \in [1, q_0]$.

- *Case 1*: $i = j$, set $\mathsf{id}_j = \mathsf{id}_A$ and add $(\mathsf{id}_A, g^a, *)$ to L_0, then return g^a.
- *Case 2*: $i \neq j$ and $(\mathsf{id}_i, g^r, r) \in L_0$, return g^r.
- *Case 3*: $i \neq j$ and $(\mathsf{id}_i, g^r, r) \notin L_0$, choose random $r \in Z_p^*$ and add (id_i, g^r, r) to L_0, then return g^r.

(2) H_1 *oracle*: $R\|m$

- *Case 1*: $((R\|m), h_1) \in L_1$, return h_1.
- *Case 2*: $((R\|m), h_1) \notin L_1$, chooses random $h_1 \in Z_p^*$ and add $(R\|m, h_1)$ to L_1, then returns h_1.

(3) H_2 *oracle*: T

- *Case 1*: $(T, h_2) \in L_2$, return h_2.
- *Case 2*: $(T, h_2) \notin L_2$, choose random $h_2 \in \{0,1\}^{k_0+k_1+n}$ and add (T, h_2) to L_2, then return h_2.

(4) Extract queries: id_i

- *Case 1*: $\mathsf{id} = \mathsf{id}_A$, $\mathcal{C}$ aborts.
- *Case 2*: $\mathsf{id} \neq \mathsf{id}_A$ and $(\mathsf{id}_i, g^r, r) \in L_0$, return y^r.

- *Case 3*: $\mathsf{id} \neq \mathsf{id}_A$ and $(\mathsf{id}_i, g^r, r) \notin L_0$, choose random $r \in Z_p^*$ and add (id_i, g^r, r) to L_0, then return y^r.

(5) Signcrypt queries: $(m, \mathsf{id}_s, \mathsf{id}_{dr})$

- *Case 1*: $\mathsf{id}_s \neq \mathsf{id}_A$

  (a) Choose random $r \in Z_p$.
  (b) Compute $R = H_0(\mathsf{id}_s)^r$, and $H_0(\cdot)$ is taken from H_0 *oracle*.
  (c) Compute $h = H_1(R\|m)$, and $H_1(\cdot)$ is taken from H_1 *oracle*.
  (d) Compute $S = sk_s^{(r+h)}$, and sk_s is taken from Extract queries.
  (e) Compute $T = \hat{e}\left(sk_s^r, H_0(\mathsf{id}_{dr})\right)$, where $H_0(\cdot)$ is from H_0 *oracle*.
  (f) Compute $\alpha = H_2(T) \oplus (S\|\mathsf{id}_s\|m)$, where $H_2(\cdot)$ is from H_2 *oracle*.
  (g) Return $\delta := (R, \alpha)$.

- *Case 2*: $\mathsf{id}_s = \mathsf{id}_A$

  (a) Choose random $r, h \in Z_p$.
  (b) Compute $R = g^r / H_0(\mathsf{id}_A)^h$ and $S = y^r$, where $H_0(\cdot)$ is from H_0 *oracle*.
  (c) Record $(R\|m, h)$ to L_1.
  (d) Compute $T = \hat{e}(R, sk_{dr})$, where sk_{dr} is from Extract queries.
  (e) Compute $\alpha = H_2(T) \oplus (S\|\mathsf{id}_A\|m)$, and $H_2(\cdot)$ is taken from H_2 *oracle*.
  (f) Return $\delta := (R, \alpha)$.

(6) Unsigncrypt queries: $(\delta := (R, \alpha), \mathsf{id}_{dr})$

- *Case 1*: $\mathsf{id}_{dr} \neq \mathsf{id}_A$

  (a) Set $b \leftarrow 1$.
  (b) If $(\mathsf{id}_{dr}, g^r, r) \in L_0$, then compute $T = \hat{e}(R, sk_{dr})$, where $sk_{dr} := y^r$, else set $b \leftarrow 0$.
  (c) If $b = 1$ and $(T, h_2) \in L_2$, then compute $(S\|\mathsf{id}_s\|m) = \alpha \oplus h_2$, else set $b \leftarrow 0$.
  (d) If $b = 1$ and $(R\|m, h) \in L_1$, then set $h := H_1(R\|m)$, else set $b \leftarrow 0$.
  (e) If $b = 1$ and $(\mathsf{id}_s, g^r, r) \in L_0$, then check whether $\hat{e}(S, g) \stackrel{?}{=} \hat{e}(y, R \cdot g^{rh})$ holds or not, $b \leftarrow 1$ if it holds, set else $b \leftarrow 0$.
  (f) If $b = 1$, then return $\sigma := (m, R, S, \mathsf{id}_s)$ and add it to L_u, else set return *Rej*.

- *Case 2*: $\mathsf{id}_{dr} = \mathsf{id}_A$. Step through L_2 with entries (T, h_2) as follows. Compute $(S\|id_s\|m) = \alpha \oplus h_2$. If $\mathsf{id}_s = \mathsf{id}_A$, stop the step and go to the next entry in L_2, and then begin this step again. If $(id_s, g^r, r) \in L_0$ continue, else go to the next entry in L_2, and then begin this step again. If $(R\|m, h) \in L_1$ continue, else go to the next entry in L_2, and then restart this step again. Check that $T \stackrel{?}{=} \hat{e}\left(S/sk_s^h, H_0(\mathsf{id}_{dr})\right)$. If so continue, else stop and move to the next entry in L_2 and restart this step again. Check that $\hat{e}(S, g) \stackrel{?}{=} \hat{e}\left(y, R \cdot H_0(\mathsf{id}_s)^h\right)$. If so return $\sigma := (m, R, S, \mathsf{id}_s)$ else stop and go forward to the next entry in L_2 restart this step again.

  If no message has been returned after all the previous phases, return *Rej*.

(7) Transform queries: $(\sigma := (m, R, S, \mathsf{id}_s), \mathsf{id}_{dv})$ $\mathcal{C}$ Computes $V = \hat{e}(S, H_0(\mathsf{id}_{dv}))$, and $H_0(\cdot)$ is taken from the H_0 *oracle*, and then return $\tilde{\sigma} := (m, R, V, \mathsf{id}_s)$ to $\mathcal{A}$. If $\mathsf{id}_{dv} = \mathsf{id}_A$, then add $(\underline{m, R, V, \mathsf{id}_s}, \mathsf{id}_A)$ to the list of L_t.

(8) Transform queries: $(m', \mathsf{id}_s, \mathsf{id}_{dv})$ $\mathcal{C}$ invokes the Signcrypt queries to generate a corresponding signature $\sigma' := (m', R', S', \mathsf{id}_s)$, then computes $V' := \hat{e}(S', H_0(\mathsf{id}_{dv}))$, and $H_0(\cdot)$ is taken from the H_0 *oracle*, and returns $\tilde{\sigma}' := (m', R', V', \mathsf{id}_s)$ to $\mathcal{A}$. If $\mathsf{id}_{dv} = \mathsf{id}_A$, add $(m', R', V', \mathsf{id}_s, \mathsf{id}_A)$ to $L_{t'}$.

(9) D-Verify queries: $(\tilde{\sigma} := (m, R, V, \mathsf{id}_s), \mathsf{id}_{dv})$

- *Case 1*: $\mathsf{id}_{dv} \neq \mathsf{id}_A$

  * Initialize $b \leftarrow 1$.
  * If $(\mathsf{id}_{dv}, g^r, r) \in L_0$, then compute $sk_{dv} = y^r$, else set $b \leftarrow 0$.
  * If $b = 1$ and $(R\|m, h)$ is in L_1, check whether $V \stackrel{?}{=} \hat{e}\left(sk_{dv}, R \cdot H_0(\mathsf{id}_s)^h\right)$ holds or not, $b \leftarrow 1$ if it holds, else set $b \leftarrow 0$.
  * Return *Acc* if $b = 1$, else return *Rej*.

- *Case 2*: $\mathsf{id}_{dv} = \mathsf{id}_A$

  * If $(m, R, V, \mathsf{id}_s, \mathsf{id}_A) \in L_t$ or $(m, R, V, \mathsf{id}_s, \mathsf{id}_A) \in L_{t'}$, return *Acc*.
  * If $(m, R, V, \mathsf{id}_s, \mathsf{id}_A) \notin L_t$ and $(m, R, V, \mathsf{id}_s, \mathsf{id}_A) \notin L_{t'}$, step through L_u with

entries (m, R, S, id_s). If $V = \hat{e}(S, H_0(\mathsf{id}_A))$ for some S, return *Acc*, else return *Rej*.

- **Challenge**: After finishing the execution of phase q, $\mathcal{A}$ outputs two messages $\{m_0, m_1\}$ and two identities $\{\mathsf{id}_s^*, \mathsf{id}_{dr}^*$, which are wished to be challenged. Here, $\mathsf{id}_{dr}^* = \mathsf{id}_A$ with probability $1/q_0$, if so, $\mathcal{C}$ chooses random $\alpha^* \in \{0,1\}^{k_0+k_1+n}$ and sets $R^* = g^b$, then returns $\delta^* := (R^*, \alpha^*)$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ aborts.
- **Queries***(phase 2)*: After receiving δ^*, $\mathcal{A}$ can make more queries as in phase 1 but

  (1) id_{dr}^* is never taken as input during Extract queries.
  (2) $(\delta^*, \mathsf{id}_{dr}^*)$ is never taken as input during Unsigncrypt queries.

- **Guess**: At last, $\mathcal{A}$ presents his guess bit $b' \in \{0,1\}$ as the output.
  $\mathcal{C}$ ignores this bit and randomly chooses element T from L_2 as the solution for the given instance of BDH problem. If $\mathcal{A}$ does not make the query T to the H_2 oracle, $\mathcal{C}$ will aborts. However, if $\mathcal{A}$ has some advantage to guess the bit correctly, it implies that this query is required, and thus there are enough information in L_2 to help $\mathcal{C}$ to pick this right T with probability $1/q_2$. If so, $T = \hat{e}(R^*, sk_A) = \hat{e}(g^b, g^{ac}) = \hat{e}(g,g)^{abc}$ is just the solution, with probability $\frac{\epsilon}{q_0 q_2}$, for the given instance of BDH problem. This is contradictory to our assumption in the begin of the proof. This concludes the proof of the **Theorem 1**. □

Theorem 2 (EU-IDUDVSC-CMA)**.** *The proposed ID-UDVSC scheme is existentially unforgeable against adaptively chosen message attack if the hardness of BDH problem holds.*

Proof. We assume that there exists an EU-IDUDVSC-CMA adversary $\mathcal{A}$ wins the defined experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{eu–idudvsc–cma}}(k)$ with a non-negligible probability ϵ. We then can design a challenger, $\mathcal{C}$, who will take the adversary, $\mathcal{A}$, as its subroutine to solve the BDH problem with a non-negligible probability. The instance $(G_1, G_T, \hat{e}, p, g, g^a, g^b, g^c)$ of the BDH problem is given to the challenger $\mathcal{C}$, who will try to solve the given instance using $\mathcal{A}$ as a subroutine. That is, $\mathcal{C}$ will try to compute the value $\hat{e}(g,g)^{abc} \in G_T$.

At first, the challenger $\mathcal{C}$ sets $y := g^c$ as the system master public key and randomly chooses id_A and id_B as two identities. $\mathcal{C}$ maintains seven lists $L_0, L_1, L_2, L_s, L_u, L_t$, and $L_{t'}$ corresponding to the H_0, H_1, H_2, Signcrypt, Unsigncrypt, Transform, and $\overline{\text{Transfrom}}$ oracles, respectively. Then, $\mathcal{C}$ plays the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{eu–idudvsc–cma}}(k)$ with $\mathcal{A}$. Furthermore, we say that $\mathcal{A}$ makes q_0 times queries to H_0 *oracle*.

- **Setup:** The challenger $\mathcal{C}$ sends *params* :=< G_1, G_T, $\hat{e}$, p, g, y, H_0, H_1, H_2 > to $\mathcal{A}$, and $(G_1, G_T, \hat{e}, p, g)$ is an instance of the BDH problem, $y = g^c$. Three hash functions $H_i(\cdot)$ for $i = 0, 1, 2$, controlled by $\mathcal{C}$, are regarded as random oracles.
- **Queries**: The adversary $\mathcal{A}$ does some queries to $\mathcal{C}$ and $\mathcal{C}$ answers them as follows.

  (1) H_0 *oracle*: id_i
  At the beginning, $\mathcal{C}$ chooses random $j, k \in [1, q_0], j \neq k$.

  - *Case 1*: $i = j$, set $\mathsf{id}_j = \mathsf{id}_A$ and add $(\mathsf{id}_A, g^a, *)$ to L_0, then return g^a.
  - *Case 2*: $i = k$, set $\mathsf{id}_k = \mathsf{id}_B$ and add $(\mathsf{id}_B, g^b, *)$ to L_0, then return g^b.
  - *Case 3*: $i \neq j, i \neq k$ and $(id_i, g^r, r) \in L_0$, return g^r.
  - *Case 4*: $i \neq j, i \neq k$ and $(id_i, g^r, r) \notin L_0$, choose random $r \in Z_p^*$ and add (id_i, g^r, r) to L_0, then return g^r.

  (2) H_1 and H_2 *oracles* are the same as in the proof of the Theorem 1.
  (3) Extract queries: id_i

  - *Case 1*: $\mathsf{id}_i = \mathsf{id}_A$ or $\mathsf{id}_i = \mathsf{id}_B$, $\mathcal{C}$ aborts.
  - *Case 2*: $\mathsf{id}_i \neq \mathsf{id}_A, \mathsf{id}_i \neq \mathsf{id}_B$ and $(\mathsf{id}_i, g^r, r) \in L_0$, return y^r.
  - *Case 3*: $\mathsf{id}_i \neq \mathsf{id}_A, \mathsf{id}_i \neq \mathsf{id}_B$ and $(\mathsf{id}_i, g^r, r) \notin L_0$, choose random $r \in Z_p^*$ and add (id_r, g^r, r) to L_0, then return y^r.

  (4) Signcrypt queries: $(m, \mathsf{id}_s, \mathsf{id}_{dr})$

  - *Case 1*: $\mathsf{id}_s \neq \mathsf{id}_A$ and $\mathsf{id}_s \neq \mathsf{id}_B$
    Use the simulation of the case 1 of the proof of the Theorem 1.
  - *Case 2*: $\mathsf{id}_s = \mathsf{id}_A, \mathsf{id}_{dr} \neq \mathsf{id}_A$ and $\mathsf{id}_{dr} \neq \mathsf{id}_B$
    Use the simulation of the case 2 of the proof in the Theorem 1.
  - *Case 3*: $\mathsf{id}_s = \mathsf{id}_B, \mathsf{id}_{dr} \neq \mathsf{id}_A$ and $\mathsf{id}_{dr} \neq \mathsf{id}_B$
    Use the simulation of the case 2 to replace id_A with id_B.
  - *Case 4*: $\mathsf{id}_s = \mathsf{id}_A$ and $\mathsf{id}_{dr} \neq \mathsf{id}_B$

    * Follow the three steps of case 2.
    * Choose random $h_2 \in \{0,1\}^{k_0+k_1+n}$.
    * Compute $\alpha = h_2 \oplus (S \| \mathsf{id}_A \| m)$.
    * Record $(\mathsf{id}_A, \mathsf{id}_B, R, \alpha, S, m, r, h, h_2)$ to L_s.
    * Return $\delta := (R, \alpha)$.

- *Case 5*: $\mathsf{id}_s = \mathsf{id}_B$ and $\mathsf{id}_{dr} \neq \mathsf{id}_A$
  Use the simulation of case 4 swapping id_A with id_B.

(5) Unsigncrypt queries: $(\delta := (R, \alpha), \mathsf{id}_{dr})$

- *Case 1*: $\mathsf{id}_{dr} \neq \mathsf{id}_A$ and $\mathsf{id}_{dr} \neq \mathsf{id}_B$
  Use the simulation of the case 1 of the proof in the Theorem 1.
- *Case 2*: $\mathsf{id}_{dr} = \mathsf{id}_B$

  * If $(\mathsf{id}_A, \mathsf{id}_B, R, \alpha, S, m, r, h, h_2) \in L_s$, return $\sigma := (m, R, S)$.
  * Otherwise, record $(\delta := (R, \alpha), \mathsf{id}_{dr})$ to L_u and go through L_2 together with entries (T, h_2).

    · Compute $(S \| \mathsf{id}_s \| m) = \alpha \oplus h_2$.
    · If $\mathsf{id}_s = \mathsf{id}_A$ or $\mathsf{id}_s = \mathsf{id}_B$, stop and go forward to the next entry in L_2, and restart this step again.
    · If $(\mathsf{id}_s, g^r, r) \in L_0$ continue, else go forward to the next entry in L_2, and restart this step again.
    · If $(R \| m, h) \in L_1$ continue, else move to the next entry in L_2, and restart this step again.
    · Check that $T \stackrel{?}{=} \hat{e}\left(S/sk_s^h, H_0(\mathsf{id}_{dr})\right)$. If so continue, else stop and go forward to the next entry in L_2, and restart this step again.
    · Check that $\hat{e}(S, g) \stackrel{?}{=} \hat{e}\left(y, R \cdot H_0(\mathsf{id}_s)^h\right)$. If so return $\alpha := (m, R, S, \mathsf{is}_s)$, else stop and go forward to the next entry in L_2 and restart this step again.

  * If there is no message that has been returned after stepping through L_2, and stepping through L_s with its entries as follows.

    · If there are entries $(\mathsf{id}_A, \mathsf{id}_B, R', \alpha, S, m', r, h, h_2)$ from L_s, then check that $R' \stackrel{?}{=} R$. If so, then it continues, else stop and go forward to the next entry of L_s, and restart this step again.
    · If there exists entry $(\mathsf{id}_B, \mathsf{id}_A, R', \alpha, S, m', r, h, h_2) \in L_s$, check that $\hat{e}(R', H_0(\mathsf{id}_A)) \stackrel{?}{=} \hat{e}(R, H_0(\mathsf{id}_B))$. If so, then it continues, else go forward to the next entry of L_s, and restart this step again.
    · Compute $(S \| \mathsf{id}_s \| m) = \alpha \oplus h_2$.
    · If $\mathsf{id}_s = \mathsf{id}_B$, stop and go forward to the next entry of L_s and restart this step again.
    · If $(\mathsf{id}_s, g^r, r) \in L_0$ continue, else go forward to the next entry of L_s and restart this step again.
    · If $(R \| m, h) \in L_1$ continue, else go forward to the next entry of L_s and restart this step again.
    · Check that $\hat{e}(S, g) \stackrel{?}{=} \hat{e}(y, R \cdot g^{rh})$. If so return $\sigma := (m, R, S, \mathsf{id}_s)$, else go forward to the next entry of L_s and restart this step again.

  * If there is no message that has been returned after all previous phases, then return *Rej*.

(6) Transform queries: $(\sigma := (m, R, S, \mathsf{id}_s), \mathsf{id}_{dv})$
$\mathcal{C}$ computes $V = \hat{e}(S, H_0(\mathsf{id}_{dv}))$, and $H_0(\cdot)$ is taken from the H_0 *oracle*, then return $\tilde{\sigma} := (m, R, V, \mathsf{id}_s)$. If $\mathsf{id}_{dv} = \mathsf{id}_A$ or $\mathsf{id}_{dv} = \mathsf{id}_B$, add $(m, R, V, \mathsf{id}_s, \mathsf{id}_A)$ or $(m, R, V, \mathsf{id}_s, \mathsf{id}_B)$ to L_t.

(7) Transform queries: $(m', \mathsf{id}_s, \mathsf{id}_{dv})$
$\mathcal{C}$ invokes the Singcrypt queries to generate a corresponding signature $\sigma' := (m', R', S', \mathsf{id}_s)$, then computes $V' = \hat{e}(S', H_0(\mathsf{id}_{dv}))$, and $H_0(\cdot)$ is taken from the H_0 oracle, and return $\tilde{\sigma}' := (m', R', V', \mathsf{id}_s)$. If $\mathsf{id}_{dv} = \mathsf{id}_A$ or $\mathsf{id}_{dv} = \mathsf{id}_B$, add $(m', R', V', \mathsf{id}_s, \mathsf{id}_A)$ or $(m', R', V', \mathsf{id}_s, \mathsf{id}_B)$ to $L_{t'}$.

(8) D-Verify queries: $(\tilde{\sigma} := (m, R, V, \mathsf{id}_s), \mathsf{id}_{dv})$

- *Case 1*: $\mathsf{id}_{dv} \neq \mathsf{id}_A$ and $\mathsf{id}_{dv} \neq \mathsf{id}_B$

* Initialize $b \leftarrow 1$.
* If $(\mathsf{id}_{dv}, g^r, r) \in L_0$, then compute $sk_{dv} = y^r$, else set $b \leftarrow 0$.
* If $b = 1$ and $(R\|m, h)$ is in L_1, check whether $V \stackrel{?}{=} \hat{e}\left(y^r, R \cdot H_0(\mathsf{id}_s)^h\right)$ holds or not, $b \leftarrow 1$ if it holds, else set $b \leftarrow 0$.
* Return *Acc* if $b = 1$, else return *Rej*.

- *Case 2*: $\mathsf{id}_{dv} = \mathsf{id}_A$ or $\mathsf{id}_{dv} = \mathsf{id}_B$

* If $(m, R, V, \mathsf{id}_s, \mathsf{id}_{dv}) \in L_t$ or $(m, R, V, \mathsf{id}_s, \mathsf{id}_{dv}) \in L_{t'}$, return *Acc*.
* If $(m, R, V, \mathsf{id}_s, \mathsf{id}_{dv}) \notin L_t$ and $(m, R, V, \mathsf{id}_s, \mathsf{id}_{dv}) \notin L_{t'}$, step through L_u with entries (m, R, S, id_s). If $V = \hat{e}(S, H_0(\mathsf{id}_{dv}))$ for some S, return *Acc*, else return *Rej*.

- **Output**: At last, the adversary $\mathcal{A}$ outputs a valid transformational signature $\tilde{\sigma}^* := \left(m^*, R^*, V^*, \mathsf{id}_s^*\right)$ together with a designated verifier id_{dv}^* by the probability ϵ. If $\left\{\mathsf{id}_s^*, \mathsf{id}_{dv}^*\right\} \neq \{\mathsf{id}_A, \mathsf{id}_B\}$, $\mathcal{C}$ aborts. Otherwise, $\left\{\mathsf{id}_s^*, \mathsf{id}_{dv}^*\right\} = \{\mathsf{id}_A, \mathsf{id}_B\}$ with probability $1/\binom{q_0}{2} = 2/q_0(q_0-1)$. Then, $\mathcal{C}$ repeats the same random tape; however, $\mathcal{C}$ uses the different choice from a random set for H_1 *oracle*. This is as done in the *forking lemma* [29]. $\mathcal{C}$ can obtain another valid transformational signature $\tilde{\sigma}^{*'} := \left(m^*, R^*, V^{*'}, \mathsf{id}_s^*\right)$ with the same designated verifier id_{dv}^*, where $\mathcal{A}$ takes $\left(R^*\|m^*\right)$ as input to the H_1 *oracle* twice; however, $\mathcal{C}$ returns two different values $h^* \neq h^{*'}$. Finally, $\mathcal{C}$ can compute

$$
\begin{aligned}
&(V^*/V^{*'})^{(h^*-h^{*'})^{-1}} \pmod p \\
&= \frac{\hat{e}\left(sk_B, \left(R^* \cdot H_0\left(\mathsf{id}_A\right)^{h^*}\right)\right.}{\left.\left(R^* \cdot H_0\left(\mathsf{id}_A\right)^{h^{*'}}\right)\right)^{\left(h^*-h^{*'}\right)^{-1}} \pmod p} \\
&= \hat{e}\left(sk_B, H_0\left(\mathsf{id}_A\right)^{\left(h^*-h^{*'}\right)^{-1}\left(h^*-h^{*'}\right)^{-1}} \pmod p\right) \\
&= \hat{e}\left(g^{bc}, g^a\right) \\
&= \hat{e}(g,g)^{abc},
\end{aligned}
$$

with the probability $\frac{2\epsilon}{q_0(q_0-1)}$. This is contradictory to our assumption in the beginning of the proof. This concludes the proof of the **Theorem 2**.

□

Theorem 3 (NT-IDUDVSC-CMA)**.** *The proposed ID-UDVSC scheme is non-transferable against adaptively chosen message attack.*

Proof.

- **Setup:** The challenger $\mathcal{C}$ runs the setup algorithm to generate the system parameters and the master key, *params* and *msk*, respectively, then sends the following information *params* :=< G_1, G_T, $\hat{e}$, p, g, y, H_0, H_1, H_2 > to $\mathcal{D}$. Three hash functions $H_i(\cdot)$ for $i = 0, 1, 2$, controlled by $\mathcal{C}$, are regarded as random oracles.
- **Queries***(phase 1)*: $\mathcal{D}$ does some queries to $\mathcal{C}$ and $\mathcal{C}$ answers them as follows.

(1) H_i *oracle*: $\mathcal{C}$ maintains L_i-*List*, where $i = 0, 1, 2$. Initially, L_i is an empty list. When $\mathcal{A}$ takes corresponding queries $\cdot$ as input to H_i *oracle*, $\mathcal{C}$ searches the L_i-*List*.

- *Case 1*: $(\cdot, h_i) \in L_i$, return h_i.
- *Case 1*: $(\cdot, h_i) \notin L_i$, choose random h_i and add $(\cdot, h_i)$ to the corresponding L_i-*List*, then return h_i to $\mathcal{D}$.

(2) Because $\mathcal{C}$ has the master key and controls the hash oracles, he can run Extract, Signcrypt, Unsigncrypt, Transform, $\overline{\text{Transform}}$, D-Verify algorithms, so he can answer these corresponding queries, respectively.

- **Challenge**: if $\mathcal{D}$ finishes the execution of phase 1, $\mathcal{D}$ submits $(m^*, \mathsf{id}_s^*, \mathsf{id}_{dv}^*)$ to $\mathcal{C}$ as the challenge. Then, $\mathcal{C}$ randomly chooses a bit b from $\{0, 1\}$ and does

(1) If $b = 1$, run the Transform algorithm to generate a $\tilde{\sigma}^* := (m^*, R^*, V^*, \mathsf{id}_s^*)$, then return it.

(2) If $b = 0$, run the $\overline{\text{Transform}}$ algorithm to generate a $\tilde{\sigma}^{*'} := (m^*, R^{*'}, V^{*'}, \mathsf{id}_s^*)$, then return it.

- **Queries***(phase 2)*: After receiving $\tilde{\sigma}$ or $\tilde{\sigma}'$, $\mathcal{D}$ can make more queries as in phase 1.
- **Guess**: At last, $\mathcal{D}$ outputs his guess bi $b' \in \{0, 1\}$.

Now, we show that the transformational signature $\tilde{\sigma}^{*'}$, which is generated by the $\overline{\text{Transfrom}}$ algorithm, is indistinguishable from $\tilde{\sigma}^*$, which is also generated by the Transform algorithm. In other words, the following distributions are identical: for $r \stackrel{R}{\leftarrow} Z_p$,

$$
\tilde{\sigma}^* = \left(V^*, R^*\right) : \begin{cases} V^* = \hat{e}\left(S^*, H_0\left(\mathsf{id}_{dv}^*\right)\right) \\ \quad = \hat{e}\left(sk_s^{*(r+h)}, H_0\left(\mathsf{id}_{dv}^*\right)\right) \\ h = H_1\left(R^*\|m^*\right) \\ R = H_0\left(\mathsf{id}_s^*\right)^r \end{cases}
$$

and

$$\tilde{\sigma}^{*'} = \left(V^{*'}, R^{*'}\right) : \begin{cases} V^{*'} = \hat{e}\left(sk^{*}_{dv}, R^{*'} \cdot H_0\left(\mathrm{id}^{*}_{s}\right)^{h}\right) \\ h = H_1\left(R^{*'} \| m^{*}\right) \\ R^{*'} \overset{R}{\leftarrow} G_1. \end{cases}$$

Therefore, we have

$$\mathbf{Pr}\left[\tilde{\sigma}^{*} = \tilde{\sigma}^{*'}\right] = \mathbf{Pr}\left[\begin{array}{c} V^{*} = V^{*'} \\ R^{*} = R^{*'} \end{array}\right] = \mathbf{Pr}\left[R^{*} = R^{*'}\right] = 1/p$$

and

$$\begin{aligned} \mathbf{Pr}\left[\tilde{\sigma}^{*'} = \tilde{\sigma}^{*}\right] &= \mathbf{Pr}\begin{array}{c} V^{*'} = V^{*} \\ R^{*'} = R^{*} \end{array} = \mathbf{Pr}\left[R^{*'} = R^{*}\right] \\ &= 1/|G_1| = 1/p, \end{aligned}$$

which means that both of them have the same probability of distributions. So, although the distinguisher $\mathcal{D}$ can make queries to the challenger $\mathcal{C}$, he cannot obtain any information to distinguish $\tilde{\sigma}^{*'}$ from $\tilde{\sigma}^{*}$. This concludes the proof of Theorem 3. □

6. CONCLUSION

We proposed the new concept of UDVSC and constructed a concrete identity-based UDVSC scheme in this paper. In our scheme, the privacy of signature holder can be guaranteed even in a public transmission channel without secure channel, and it improves the practical applicability of normal UDVSs. The security proofs for our proposed ID-UDVSC scheme are given in the random oracle model. How to design the concrete scheme, which can be provably secure in the standard model (without the assumption of random oracle), is still an open problem.

ACKNOWLEDGEMENTS

This research is supported in part by the National Natural Science Foundations of China under Grants No. 61103247, 61102093 and 61072080, the Natural Science Foundation of Fujian Province under Grant No. 2011J05147, and the Foundation for Excellent Young Teachers of Fujian Normal University (No.fjsdjk2012049).

REFERENCES

1. Jakobsson M, Sako K, Impagliazzo R. Designated verifier proofs and their applications. In *Advances in Cryptology-Eurocrypt'96*, vol. 1070, LNCS. Springer-Verlag: Saragossa, Spain, 1996; 143–154.
2. Steinfeld R, Bull L, Wang H, Pieprzyk J. Universal designated-verifier signatures. In *Advances in Cryptology-Asiacrypt'03*, vol. 2894, LNCS. Springer-Verlag: Taipei, Taiwan, 2003; 523–542.
3. Huang X, Susilo W, Mu Y, Zhang F. Short (identity-based) strong designated verifier signature scheme. In *Proceedings of ISPEC'06*, vol. 3903, LNCS. Springer-Verlag: Hangzhou, China, 2006; 214–225.
4. Zheng Y. Digital signcryption or how to achieve cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). In *Advances in Cryptology - Crypto'97*, vol. 1294, LNCS. Springer-Verlag: Santa Barbara, California, 1997; 165–179.
5. Shamir A. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology-Crypto'84*, vol. 196, LNCS. Springer-Verlag: Santa Barbara, California, 1985; 47–53.
6. Chen L, Lee JM. Improved identity-based signcryption. In *Public Key Cryptography-PKC'05*, vol. 3386, LNCS. Springer-Verlag: Les Diablerets, Switzerland, 2005; 362–379.
7. Steinfeld R, Wang H, Pieprzyk J. Efficient extension of standard Schnorr/RSA signatures into universal designated-verifier signatures. In *Public Key Cryptography-PKC'04*, vol. 3089, LNCS. Springer-Verlag: Singapore, 2004; 86–100.
8. Ng C Y, Susilo W, Mu Y. Universal designated multi verifier signature schemes, *Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems*, Fukuoka, Japan. IEEE Press, 2005; 305–309.
9. Shahandashti S F, Safavi-Naini R. Generic constructions for universal designated-verifier signatures and identity-based signatures from standard signatures. *IET Information security* 2009; **3**(4): 152–176.
10. Zhang R, Furukawa J, Imai H. Short signature and universal designated verifier signature without random oracles. In *Proceedings of ACNS'05*, vol. 3531, LNCS. Springer-Verlag: New York, USA, 2005; 483–498.
11. Boneh D, Boyen X. Short signatures without random oracles. In *Advances in Cryptology-Eurocrypt'04*, vol. 3027, LNCS. Springer-Verlag: Interlaken, Switzerland, 2004; 514–532.
12. Laguillaumie F, Libert B, Quisquater JJ. Universal designated verifier signatures without random oracles or non-black box assumptions. In *Proceedings of the Fifth Conference on Security and Cryptography for Networks*, vol. 4116, LNCS. Springer-Verlag: Maiori, Italy, 2006; 63–77.
13. Huang X, Susilo W, Mu Y, Wu W. Secure universal designated verifier signature without random oracles.

International Journal of Information Security 2008; **7**: 171–183.

14. Vergnaud D. New extensions of pairing-based signatures into universal (multi) designated verifier signatures. In *Proceedings of 33rd International Colloquium on Automata, Languages and Programming-ICALP'06*, vol. 4052, LNCS. Springer-Verlag: Venice, Italy, 2006; 58–69.
15. Zhang F, Susilo W, Mu Y, Chen X. Identity-based universal designated verifier signatures. In *Proceedings of EUC Workshops'05*, vol. 3823, LNCS. Springer-Verlag: Nagasaki, Japan, 2005; 825–834.
16. Cao F, Cao Z. An identity based universal designated verifier signature scheme secure in the standard model. *Journal of Systems and Software* 2009; **82**: 643–649.
17. Waters B. Efficient identity-based encryption without random oracles. In *Advances in Cryptology-Eurocrypt'05*, vol. 3494, LNCS. Springer-Verlag: Aarhus, Denmark, 2005; 114–127.
18. Seo S H, Hwang J Y, Choi K Y, Lee D H. Identity-based universal designated multi-verifiers signature schemes. *Computer Standards and Interfaces* 2008; **30**: 288–295.
19. Baek J, Safavi-Naini R, Susilo W. Universal designated verifier signature proof (or how to efficiently prove knowledge of a signature). In *Advances in Cryptology-Asiacrypt'05*, vol. 3788, LNCS. Springer-Verlag: Aarhus, Denmark, 2005; 644–661.
20. Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. In *Advances in Cryptology-Asiacrypt'01*, vol. 2248, LNCS. Springer-Verlag: Gold Coast, Australia, 2001; 566–582.
21. Chen X, Chen G, Zhang F, Wei B, Mu Y. Identity-based universal designated verifier signature proof system. *International Journal of Network Security* 2009; **8**(1): 52–58.
22. Li Y, Pang L, Wang Y. Attacks on a universal designated verifier signature scheme, *In the Fifth International Conference on Information Assurance and Security*, Xi'an, China. IEEE Press, 2009; 27–30.
23. Huang X, Susilo W, Mu Y, Zhang F. Restricted universal designated verifier signature. In *Proceedings of UIC'06*, vol. 4159, LNCS. Springer-Verlag: Wuhan and Three Gorges, China, 2006; 874–882.
24. Laguillaumie F, Vergnaud D. On the soundness of restricted universal designated verifier signature and dedicated signatures. In *Proceedings of ISC'07*, vol. 4779, LNCS. Springer-Verlag: Valparaíso, Chile, 2007; 175–188.
25. Huang X, Susilo W, Mu Y, Wu W. Universal designated verifier signature without delegatability. In *Proceedings of ICICS'06*, vol. 4307, LNCS. Springer-Verlag: Raleigh, NC, USA, 2006; 478–498.
26. Chang T. An ID-based multi-signer universal designated multi-verifier signature scheme. *Information and Computation* 2011; **209**: 1007–1015.
27. Li J, Wang Y. Universal designated verifier ring signature (proof) without random oracles. In *Proceedings of EUC Workshops'06*, vol. 4097, LNCS. Springer-Verlag: Seoul, Korea, 2006; 332–341.
28. Boneh D, Franklin M. Identity-based encryption from the Weil pairing. In *Advances in Cryptology-Crypto'01*, vol. 2139, LNCS. Springer-Verlag: Santa Barbara, California, 2001; 213–229.
29. Pointcheval D, Stern J. Security arguments for digital signatures and blind signatures. *Journal of Cryptology* 2000; **13**: 361–369.