A novel certificateless deniable authentication protocol

Chunhua Jin, Chunxiang Xu, Xiaojun Zhang, Qianna Xie, Fagen Li

School of Computer Science and Engineering,

University of Electronic Science and Technology of China, Chengdu, 611731, China

E-mail:chunhuaking@gmail.com

Abstract: Deniable authenticated protocol is a new and attractive protocol compared to the traditional authentication protocol. It allows the appointed receiver to identify the source of a given message, but not to prove the identity of the sender to a third party even if the appointed receiver is willing to reveal its private key. In this paper, we first define a security model for certificateless deniable authentication protocols. Then we propose a non-interactive certificateless deniable authentication protocol, by combining deniable authentication protocol with certificateless cryptography. In addition, we prove its security in the random oracle model.

Keywords: Deniable authentication, Certificateless cryptography, The random oracle model

1 Introduction

Nowadays, authentication has emerged to be an essential communication process. The aim of this process is to ensure the validity of the parties involved in a communication system. In some communication systems, digital signature can provide such authentication. In a digital signature, the private key of the signer is tied to it as well as the message being signed. The signature can be verified easily by using the public key of the signer in the verification phase. The verifier (even any eavesdropper) can identify the source of a given message and provide the signer's identity proof to any third party as well. Hence, the signer will not be able to deny its participation in this communication. Generally, this notion is known as non-repudiation. However, in other communication systems, the non-repudiation property is undesirable. Such as electronic voting systems, online shopping and secure negotiations over the Internet [1]. In an electronic voting system, let A be a voter and B be a voting center. Suppose a third party C compels A to elect a candidate. However, the voter A does not intend to elect the candidate. A is made to cast its ballot m as well as the authenticator to the voting center B so that B can ensure that this ballot is from A but not from anyone else. In addition, B cannot prove the ballot m to C even if B fully cooperates with C. If there is full cooperation between them, yet C may be sceptical of the truth of the evidence given by B. Thus, C cannot force A to select the candidate because A can deny that it sent B the ballot. Hence, in order to protect the voter from coercion in electronic voting systems, we need a protocol which enables a receiver to identify the source of a given message, but not prove to a third party the identity of the sender. In an online shopping system, let C be a customer and M be a merchant. Suppose that C wants to order goods from M, it will bargain with M. After several bargaining, M will finally make a favorable price m to C. Whereas, for the benefit of M, M will not expect the customer C to show this favorable price to other customers. Therefore, in an online shopping system, it needs such a special requirement: the customer C can identify the source of a given favorable price m, but cannot prove to any other customers the identity of the sender M. These instances show that the deniable authentication protocol is very important. It mainly has two characteristics: (1) it enables a assigned receiver to identify the source of a given message; (2) the assigned receiver can not prove the source of a given message to a third party even if the receiver reveals its own private key to the third party. Hence, it plays a very important part in practice. It is imperative for us to design such a protocol. In recent years, many related protocols [2, 3, 4, 5, 6, 7, 8, 9, 10] have been proposed. However, these protocols are based on public key infrastructure (PKI). PKI may bring some problems, such as certificate generation, distribution, storage and revocation which impede the development of PKI.

To simplify key management and avoid the use of public key certificates, identity-based (ID-based) cryptography was introduced by Shamir in 1984 [11]. In an ID-based system, a user can use a binary string which can uniquely identify the user as its public key, such as telephone number, email address, etc. The associated private key is generated by a trusted party called private key generator

(PKG). The PKG is responsible for generating the user's private key by inputting its identity and the master private key, which is owned by the PKG. The user's public key is just its identity, and there is no need to use public key certificates. Therefore, in order to reduce the communication cost and improve the communication efficiency, a number of ID-based deniable authentication protocols have be presented [12, 13, 14, 15, 16, 17]. However, there is a basic assumption that the PKG is unconditional trustable in ID-based cryptography. This is because the PKG can get every user's private key in this system. Therefore, ID-based cryptography suffers from the key escrow problem.

In order to solve the key escrow problem which is the inherent issue of ID-based cryptography, Al-Riyami and Paterson [18] proposed a new paradigm called certificateless public-key cryptography (CL-PKC) in 2003. In a CL-PKC, a user's full private key is not generated by the key generation center (KGC) alone. Instead, a user combines its partial private key produced by the KGC with some secret information produced by the user itself to create its full private key. In this way, a user's private key is not available to the KGC, and its public key is also generated by combining its secret information with the KGC's public parameters. The system is not ID-based, because the public key is no longer computable from an identity (or identifier) alone. Up to now, there has been no certificateless deniable authentication (CL-DA) protocol. Therefore, it is imperative for us to devise a provable secure deniable authentication protocol based on certificateless cryptography.

1.1 Related works

In 1998, Dwork et al. [2] developed a protocol based on concurrent zero-knowledge proof. However, the protocol requires a timing constraint and the proof of knowledge is time-consuming. Aumann and Rabin [3] also proposed another protocol based on the factoring problem in the same year. Nevertheless, their protocol needs a pubic directory trusted by the sender and the receiver. In 2001, Deng et.al [4] presented two deniable authentication protocols, which were based on the factoring problem and the discrete logarithm problem, respectively. However, these protocols also requires a trusted public directory. To overcome this problem, Fan et.al [5] proposed a new deniable authenticated protocol based on the Diffie-Hellman key distribution protocol in 2002. Whereas, in 2005, Yoon et al. [6] pointed out that their protocol suffered from the intruder masqueradeing attack. Then they proposed an enhanced deniable authentication protocol based on Fan et al.'s protocol. Yet it is still an interactive protocol. Subsequently, many interactive protocols have been proposed [7, 12, 13, 14]. However, protocols [12, 13] can not resist key compromise impersonation (KCI) attack. The KCI attack means known-key attack. An adversary can implement it after compromising a protocol entity's private key.

Since the communication cost of non-interactive deniable authentication protocol is lower than interaction deniable authentication protocol, there is a desire to design secure and efficient non-interactive deniable authentication protocol for researchers. In recent years, a lot of non-interactive deniable authentication protocols have also been proposed [8, 15, 16, 9, 10, 17]. In 2004, Shao et al. [8] proposed an efficient non-interactive deniable authentication protocol based on generalized ElGamal signature scheme. However, if a session key of the communication parties is compromised, the receiver cannot identify the true source of a forged message. In 2005, Shi et al. [15] proposed a non-interactive ID-based deniable authentication protocol from pairings. Nevertheless, it is low-efficiency because an ID-based signature scheme is used to sign a session key. Cao et al. [16] also proposed a non-interactive ID-based deniable authentication protocol using pairings. But it can not resist KCI attack. A common weakness of above protocols is lack of formal security proof which is of great importance for protocol design. In 2009, Wang et al. [9] defined a formal security model for non-interactive deniable authentication protocol. Then they presented a non-interactive deniable authentication protocol based on designated verifier proofs and proved its security in this model. In 2011, Tian et al. [10] put forward a non-interactive deniable authentication protocol. They defined a security model for non-interactive deniable authentication protocols and proved its security in this model. In 2013, Li et al. [17] proposed an efficient and non-interactive ID-based deniable authentication protocol using bilinear pairings. They defined a formal security model and proved the protocol is secure in the random oracle model.

1.2 Our Contribution

In this paper, we first define a formal security model for the non-interactive deniable authentication protocol based on certificateless cryptography. This model captures the notion of deniability and authentication of CL-DA protocol. Then we propose an efficient and non-interactive CL-DA protocol and prove its security in the random oracle model. Our protocol comes from Zhang et al.'s certificateless public key signature [19] and the spirit of our protocol is aroused by Li et al.'s ID-based deniable authentication protocol [17].

1.3 Organization of this paper

The rest of this paper is organized as follows. In the next section, we will describe some basic properties of bilinear pairings and the related hard problems. In Section 3, a formal security model for CL-DA is given. We propose an efficient and non-interactive CL-DA protocol based on bilinear pairings in Section 4. The proposed protocol is analyzed in Section 5. In Section 6, we conclude the paper.

2 Preliminaries

In this section, we introduce the basic properties of bilinear pairings, the computational Diffie-Hellman problem (CDHP) and the bilinear Diffie-Hellman problem (BDHP).

Let G_1 be an additive group and G_2 be a multiplicative group. P is the generator of G_1. They have the same large prime order q. A bilinear pairing is a map $e : G_1 \times G_1 \rightarrow G_2$ with the following properties:

(1)Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1, a, b \in \mathbb{Z}_q^*$.

(2)Non-degeneracy: There exists $P, Q \in G_1$ such that $e(P, Q) \neq 1$.

(3)Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

A bilinear map satisfying the above three properties is said to be an admissible bilinear map. The Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties can be modified to create such bilinear maps. A more details can be referred to [20].

Definition 1. Computational Diffie-Hellman Problem (CDHP): Let G_1 be an additive circle group generated by P, whose order is a prime q. The computational Diffie-Hellman problem is to compute abP given (P, aP, bP) with $a, b \in \mathbb{Z}_q^*$. The (t, ϵ)-CDH assumption holds in G_1 if no t-polynomial time adversary A has advantage at least ϵ in solving the CDH problem.

Definition 2. **Bilinear Diffie-Hellman Problem (BDHP)**: Let G_1 and G_2 be two circle groups which have the same prime order q. P is the generator of G_1, and e: $G_1 \times G_1 \rightarrow G_2$ is a bilinear map. The bilinear Diffie-Hellman problem is to compute $e(P, P)^{abc}$ given (P, aP, bP, cP) with $a, b, c \in \mathbb{Z}_q^*$. The (t, ϵ)-BDH assumption holds in (G_1, G_2, e) if no t-polynomial time adversary A has advantage at least ϵ in solving the BDH problem.

3 Formal definition for CL-DA protocol

3.1 Framework of CL-DA protocol

A CL-DA protocol consists of the following seven algorithms:

Setup: It takes as input a security parameter k, and returns system parameters *params* and a master private key s, where *params* are the global public parameters for the system, while s is only known to the KGC. It also defines a message space $\mathcal{M}$.

Partial-Private-Key-Extract: It takes as input *params*, s and an arbitrary $ID_i \in \{0, 1\}^*$, and returns the corresponding partial private key D_i which is assumed to be distributed securely to the corresponding user.

Set-Secret-Value: It takes as input *params* and a user's identity ID_i, and returns a secret value x_i.

Set-Private-Key: It takes as input *params*, a user's partial private key D_i and its secret value x_i, and returns the full private key SK_i.

Set-Public-Key: It takes as input *params* and a user's secret value x_i, and returns a public key PK_i which is publicly known.

Authenticate: It takes as input a message m, the receiver's identity ID_B, public key PK_B, the sender's identity ID_A, public key PK_A and full private key SK_A, and returns a deniable authenticator σ.

Verify: It takes as input *params*, a sender's identity ID_A, public key PK_A, the receiver's identity ID_B, public key PK_B, full private key SK_B, a message m and a deniable authenticator σ, and returns $\top$ for acceptance, or $\perp$ for rejection.

Setup and *Partial-Private-Key-Extract* are run by the KGC, but *Set-Secret-Value*, *Set-Private-*

Key and *Set-Public-Key* are run by each user. Nevertheless, *Authenticate* is run by the sender, *Verify* is run by the receiver. In order to simplify the formula, we omit all parameters *params* in later chapters and sections..

For consistency, we require that if $\sigma =$*Authenticate*$(m,\ ID_A,\ PK_A,\ SK_A,\ ID_B,\ PK_B)$, then we have $\top=$*Verify*$(m, \sigma, ID_A, PK_A, ID_B, PK_B, SK_B)$, otherwise, we have $\perp$.

3.2 Security notion

As compared with traditional authentication protocols, deniable authentication is a new authentication mechanism and mainly has the following two properties: (1) the receiver can authenticate the source of the received message; (2) it is unable to convince a third party of the sender's identity even if the receiver reveals its own private key to the third party. We define the security notion of deniable authentication protocol, borrowing the security definition of traditional digital signature scheme. Whereas, they have different security notions. Only the sender can generate a valid signature in a traditional signature scheme. In other words, no one but the sender can forge a signature for the message. In the verification phase, everyone can verify the validity of the signature because the parameters of the verification equation are open to the public. However, in a deniable authentication protocol, both the sender and the receiver can generate a valid deniable authenticator. This property is called deniability.

In certificateless cryptography, as defined in [18], there are two types of adversaries called Type I adversary and Type II adversary with different capabilities. A Type I adversary A^I does not have access to the master key, but it has the ability to replace any user's public key with a value of its choice. While a Type II Adversary A^{II} has access to the master key but cannot replace any user's public key. Since our deniable authentication protocol is based on certificateless cryptography, we must require our protocol is secure in these two types of adversaries. Here we consider two games " game I " and "game II " where A^I and A^{II} interact with their challenger in these two games, respectively. We say that a CL-DA protocol is deniable authentication against adaptive chosen message attacks (DA-CMA), if the probability of success is negligible, for any probabilistic polynomial time(PPT)adversary A^I and A^{II}.

We define two games " game I " and "game II " as follows.

Game-I(for Type I Adversary):

Setup: The challenger C runs *Setup* algorithm that takes as input a security parameter k to obtain the system parameter *params* and the master key s. C then sends *params* to the adversary A^I while keeps s secret.

Probing: The adversary A^I can perform a polynomially bounded number of following queries in an adaptive manner.

- Partial private key extraction queries: A^I can request the partial private key of a user with identity ID_i. Once receiving such a query, C computes D_i=*Partial-Private-Key-Extract*(s, ID_i) and responds it to A^I.

- Private key extraction queries: A^I can request the private key of a user whose identity is ID_i. Once receiving such a query, the challenger first computes the secret value x_i=*Set-Secret-Value*(ID_i), and then computes D_i=*Partial-Private-Key-Extract*(s, ID_i). Finally, it computes SK_i=*Set-Private-Key*$(D_i,\ x_i)$ and responds it to A^I.

- Request public key queries: A^I can request the public key of a user whose identity is ID_i. Once receiving such a query, C first computes x_i=*Set-Secret-Value*(ID_i), and then computes PK_i=*Set-Public-Key*(x_i) and responds it to A^I.

- Public key replacement queries: A^I may replace the public key PK_i with a new value chosen by it. Note that it does not require A^I to provide the corresponding secret value when making this query.

- Authenticate queries: A^I submits the requests of two identities ID_i, ID_j and a message m. Once receiving such a query, C first runs the *Set-Private-Key* to get SK_i, and then computes σ=*Authenticate*$(m,\ ID_j,\ PK_j,\ ID_i,\ PK_i,\ SK_i)$, and responds the result to A^I.

  If the public key PK_i and PK_j have been replaced by A^I, then C cannot compute SK_i and SK_j. Thus the authentication oracle's response may be wrong. In this case, we assume that A^I may additionally submit the secret information x_i' corresponding to the replaced public key PK_i' to the authentication oracle queries.

– Verify queries: A^I submits the requests of two identities ID_i, ID_j and a deniable authenticator σ, C first runs the *Set-Private-Key* to get SK_j, and then runs *Verify*$(\sigma, ID_i, PK_i, ID_j, PK_j, SK_j)$. If the result is $\top$, C responds m to A^I. Otherwise, C responds $\perp$.

Forging: Eventually, A^I outputs a tuple (m^*, σ^*, ID_i^*, ID_j^*, PK_i^*, PK_j^*), We say that A^I wins the game, if the following conditions hold:

– σ^* is a valid deniable authenticator under target identities ID_i^*, ID_j^* and the corresponding public key PK_i^*, PK_j^*.

– A^I has not request private key extraction queries for identities ID_i^*, ID_j^*.

– A^I has not request both public key replacement queries and partial private key extraction queries for identities ID_i^*, ID_j^*.

– (m^*, ID_i^*, ID_j^*, PK_i^*, PK_j^*) has never been submitted to the authenticate queries.

– (σ^*, ID_i^*, ID_j^*, PK_i^*, PK_j^*) has never been submitted to the verify queries.

The advantage of A^I is defined as the probability that it wins.

Definition 3. An adversary A^I is said to be an $(\epsilon, t, q_{par}, q_{pk}, q_{da}, q_v)$-forger of a CL-DA protocol if A^I has advantage at least ϵ in the above game, runs in time at most t, and makes at most q_{par} partial private key extraction queries, q_{pk} public key queries, q_{da} deniable authentication queries and q_v verify queries. A CL-DA protocol is said to be $(\epsilon, t, q_{par}, q_{pk}, q_{da}, q_v)$-DA-CMA secure if no $(\epsilon, t, q_{par}, q_{pk}, q_{da}, q_v)$-forger exists.

Note that the adversary A^I is not allowed to make a private key query, both a replace public key query and a partial private key query on identity ID_j^* in the above definition. This requirement is important for the deniability. Since the receiver is also able to produce a valid deniable authenticator, the sender can deny its behavior. It is the main difference between deniable authentication in CL-DA and undeniable authentication in traditional digital signature.

Game-II(for Type II Adversary):

Setup: The challenger C runs *Setup* algorithm, takes as input a security parameter k to obtain the system parameter *params* and the master key s. C then sends *params* and s to the adversary A^{II}.

Probing: The adversary A^{II} can perform a polynomially bounded number of queries in an adaptive manner. Here, we don't need partial private key queries, since A^{II} has access to the master key s and runs the partial private key queries D_i=*Partial-Private-Key-Extract*(s, ID_i) by itself.

- Private key extraction queries: A^{II} can request the private key of a user whose identity is ID_i. Once receiving such a query, C first computes D_i=*Partial-Private-Key-Extract*(s, ID_i), and then computes x_i=*Set-Secret-Value*(ID_i). Finally, it computes SK_i=*Set-Private-key* (D_i, x_i) and responds it to A^{II}.

- Request public key queries: A^{II} can request the public key queries of a user whose identity is ID_i. Once receiving such a query, C first computes x_i=*Set-Secret-Value*(ID_i), and then computes PK_i=*Set-Public-Key*$(x_i)$and responds it to A^{II}.

- Authentication queries: A^{II} submits the requests of two identities ID_i, ID_j and a message m. Once receiving such a query, C first runs the *Set-Private-Key* to get SK_i, and then computes σ=*Authenticate*$(m, ID_j, PK_j, ID_i, PK_i, SK_i)$ and responds it to A^{II}.

- Verify queries: A^{II} submits the requests of two identities ID_i, ID_j and a deniable authenticator σ, C first runs the *Set-Private-Key* to get SK_j, and then runs *Verify*$(\sigma, ID_i, PK_i, ID_j, PK_j, SK_j)$. If the result is $\top$, C responds m to A^{II}. Otherwise, C responds $\perp$.

Forging: Eventually, A^{II} outputs a tuple $(m^*, \sigma^*, ID_i^*, ID_j^*, PK_i^*, PK_j^*)$, We say that A^{II} wins the game, if the following conditions hold:

- σ^* is a valid deniable authenticator under target identities ID_i^*, ID_j^* and the corresponding public key PK_i^*, PK_j^*.

- A^{II} has not request private key extraction queries for identities ID_i^*, ID_j^*.

- $(m^*, ID_i^*, ID_j^*, PK_i^*, PK_j^*)$ has never been submitted to the authentication queries .

– (σ^*, ID_i^*, ID_j^*, PK_i^*, PK_j^*) has never been submitted to the verify queries.

The advantage of A^{II} is defined as the probability that it wins.

Definition 4. An adversary A^{II} is said to be an $(\epsilon, t, q_e, q_{pk}, q_{da}, q_v)$-forger of a CL-DA protocol if A^{II} has advantage at least ϵ in the above game, runs in time at most t, and makes at most q_e private key extraction queries, q_{pk} public key queries, q_{da} deniable authentication queries and q_v verify queries. A CL-DA protocol is said to be $(\epsilon, t, q_e), q_{pk}, q_{da}, q_v)$-DA-CMA secure if no $(\epsilon, t, q_e), q_{pk}, q_{da}, q_v)$-forger exists.

Note that the adversary A^{II} is not allowed to make a private key query on identity ID_j^* in the above definition. This term is of great importance to gain the deniability. Because the receiver can also produce a valid deniable authenticator, the sender can deny its behavior. It is the main difference between deniable authentication in CL-DA and undeniable authentication in traditional digital signature.

Definition 5. A CL-DA protocol is secure if it is DA-CMA against two types of adversaries A^I and A^{II}.

4 A New Certificateless Deniable Authentication Protocol

In this section, we propose a new certificateless deniable authentication protocol on pairings. we describe our protocol using the following seven algorithms.

Setup: On input a security parameter k, the algorithm generates (G_1, G_2, e), where G_1 and G_2 are cyclic groups of prime order q, and $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map. P is the generator of G_1. Let H_1, H_2 be two cryptographic hash functions, where H_1: $\{0,1\}^* \rightarrow G_1$ and H_2: $\{0,1\}^* \times G_1 \rightarrow \mathbb{Z}_q^*$. The KGC selects a master key $s \in \mathbb{Z}_q^*$ randomly and computes $P_{pub} = sP$. The system parameters *params*$=(G_1, G_2, e, q, P, P_{pub}, H_1, H_2)$ are publicly known and the master key s is keep secret.

Partial private key extraction: On input *params*, the master key s and a user's identity $ID_i \in \{0,1\}^*$, the KGC runs the algorithm as follows.

1. Compute $Q_i = H_1(ID_i)$.

2. Output the partial private key $D_i = sQ_i$ to the user.

Secret value extraction: On input *params* and the user's identity ID_i, the user selects a random value $x_i \in \mathbb{Z}_q^*$ and outputs x_i as its secret value.

Private key extraction: On input *params*, the user's partial private key D_i and its secret value x_i, the user outputs its full private key $SK_i = (D_i, x_i)$.

Public key extraction: On input *params*, and the user's secret value x_i, the user outputs its public key $PK_i = x_i P$.

Authenticate: On input *params*, a sender A's identity ID_A, its public key PK_A , its full private key SK_A, a receiver B's identity ID_B, its public key PK_B, and a message $m \in \{0,1\}^*$, then A follows the steps below.

1. Randomly select $r \in \mathbb{Z}_q^*$, compute $U = rQ_A$.
2. Compute $h_2 = H_2(m, U, PK_A, PK_B, x_A PK_B)$.
3. Compute $V = (r + h_2)D_A$.
4. Compute $S = e(V, Q_B)$.
5. Output a deniable authenticator $\sigma = (U, S)$.

Verify: On input *params*, the sender's identity ID_A, its public key PK_A, the receiver's identity ID_B, its public key PK_B, its full private key SK_B, and the deniable authenticator σ, then B follows the steps below.

1. Compute $h_2' = H_2(m, U, PK_A, PK_B, x_B PK_A,)$.
2. Compute $S' = e(U + h_2' Q_A, D_B)$.
3. Check whether $S' = S$. If the equation holds, output $\top$, otherwise output $\perp$.

5 Analysis of the Protocol

5.1 Security

Consistency: The consistency can be easily verified by the following equations.

$$\begin{aligned} S &= e(V, Q_B) = e((r+h_2)D_A, Q_B) \\ &= e((r+h_2)sQ_A, Q_B) = e((r+h_2)Q_A, sQ_B) \\ &= e(rQ_A + h_2Q_A, D_B) = e(U + h_2Q_A, D_B) \\ &= S' \end{aligned}$$

Deniability: After receiving the deniable authenticator $\sigma = (U, S)$, the receiver B can identify the source of a message m with its private key $SK_B = (D_B, x_B)$. To simulate the transcripts on the message m, the receiver follows the steps below.

1. Randomly select $r' \in \mathbb{Z}_q^*$ and compute $U' = r'Q_A$.
2. Compute $h'_2 = H_2(m, U', PK_A, PK_B, x_B PK_A)$.
3. Compute $S' = e(U' + h'_2 Q_A, D_B)$.

The receiver can generate $\sigma' = (U', S')$ that is indistinguishable from $\sigma = (U, S)$. σ is generated by the sender in terms of Authenticate algorithm in Section 4. Let $\overline{\sigma} = (\overline{U}, \overline{S})$ be a deniable authenticator which is randomly selected in the set of all valid sender's deniable authenticator appointed to the receiver.The probability $\Pr[\sigma' = \overline{\sigma}]$ is $1/(q-1)$ since σ' is generated from a randomly selected value $x' \in \mathbb{Z}_q^*$. Similarly, the probability $\Pr[\sigma = \overline{\sigma}]$ has the same value $1/(q-1)$ since it is generated from $x \in \mathbb{Z}_q^*$. That is to say, both of them have the same probability distribution.

Then we prove our protocol satisfies DA-CMA security in the following Theorem 1.

Theorem 1. Our proposed certificateless deniable authentication protocol is DA-CMA against type I/II adversary in the random oracle model under the BDH assumption and CDH assumption.

Proof. This theorem follows from Lemmas 1 and 2.

Lemma 1. In the random oracle model, If a probabilistic polynomial time (PPT) adversary A^I has an advantage ϵ in forging a deniable authenticator in Game I, which runs in time t, and makes q_{H_i} queries to random oracles H_i for i=1, 2, q_{par} queries to the partial private key extraction oracle, q_{pk} queries to the public key request oracle, q_{da} queries to the deniable authentication oracle, and q_v queries to the verify oracle. There exists a algorithm C that can solve the BDH problem with an advantage $\epsilon \geq 5(q_{da}+1)(q_{da}+q_{H_2})q_{H_1}/(2^k-1)$ in expected time $t' \leq 60343 q_{H_2} q_{H_1} 2^k t/\epsilon(2^k-1)$.

Proof. We use the forking lemma [21] to prove the proposed protocol. To employ the forking lemma, we need to show how our protocol fits into the signature scheme represented in [21], the simulation steps in which the deniable authentication can be simulated without the sender's private key (and thus, also without the master private key), and how we can solve BDH difficult problem based on the forgery.

First, we observe that our protocol satisfies all the required properties described in [21]. During the deniable authentication of a message m, the tuple $(\sigma_1, h_2, \sigma_2)$ is produced which corresponds to the required three-phase honest-verifier zero-knowledge identification protocol, where $\sigma_1 = U = rQ_A$ is the commitment of the prover (σ_1 can be considered to be selected randomly from a large set since r is selected randomly from $\mathbb{Z}_q^*$ and G_2 is a cyclic group of prime order q). $h_2 = H_2(m, U, PK_A, PK_B, R)$ is the hash value depending on m and σ_1 substituted for the verifier's challenge, and $\sigma_2 = S$ is the response of the prover which depends on σ_1, h_2 and the sender's partial private key D_A.

Next, we need to show a simulation step that provides a faithful simulation to the forger A^I and how to solve the BDH problem by interacting with A^I. C receives a random instance (P, aP, bP, cP) of the BDH problem. Its goal is to compute $e(P,P)^{abc}$. C will run A^I as a subroutine and act as A^I's challenger in the Type I's DA-CMA game. C needs to maintain lists L_1, L_2 which are initially empty to keep track values asked by A^I to random oracle queries H_1, H_2. Roughly speaking, these answers are randomly generated. Whereas, to avoid collision and maintain consistency for answers to these hashing oracles, C keeps two lists L_1 and L_2 respectively to store the answers.

Without loss of generality, we assume that A^I will ask for $H_1(ID_i)$ before ID_i is used in any key extraction queries, deniable authentication queries and verify queries. A^I never makes verify queries on a deniable authenticator, which is obtained from the deniable authentication queries. It just makes verify queries for observed deniable authenticators. C maintains a list $L_3 = (ID_i, D_i, PK_i, x_i)$ while A^I makes queries over all the game.

C gives A^I the system parameters with $P_{pub} = cP$. Notice that c is unknown to C. This value simulates the master private key for the KGC in the game. C responds the oracle queries of A^I as follows.

H_1 **Queries**: At first, C selects two different random numbers $a, b \in \{1, 2, \cdots, q_{H_1}\}$. A^I asks a polynomially bounded number of H_1 queries on identity of its choice. At the η-th H_1 request, C responds by $H_1(ID_\eta) = aP$. At the γ-th H_1 request, C responds by $H_1(ID_\gamma) = bP$. For requests $H_1(ID_i)$ with $i \neq \eta, \gamma$, C selects $t_i \in \mathbb{Z}_q^*$ at random, adds the tuple (ID_i, t_i) in the list L_1 and responds $H_1(ID_i) = t_iP$.

H_2 **Queries**: Suppose that A^I submits the tuple (m, U, PK_i, PK_j, R) to oracle $H_2(\cdot)$. C first checks whether H_2 has already been defined for that input. If so, C returns that defined value. Otherwise, C returns a random value $h_2 \in \mathbb{Z}_q^*$ as the answer. Then C puts the tuple $(m, U, PK_i, PK_j, R, h_2)$ into the list L_2.

Partial Private Key Queries: Suppose that the query is made on an identity ID_i. If $ID_i = ID_\eta$ or $ID_i = ID_\gamma$, C aborts. If $ID_i \neq ID_\eta, ID_\gamma$, C looks up the list L_3 and runs the algorithm as follows.

- If the list L_3 contains (ID_i, D_i, PK_i, x_i), C checks whether $D_i =\perp$. If $D_i \neq\perp$, C outputs D_i to A^I. If $D_i =\perp$, C recovers the corresponding tuple (ID_i, t_i) from the list L_1 (this means that C previously answered $H_1(ID_i) = t_iP$). The partial private key $D_i = t_iP_{pub} = t_icP$ is associated with ID_i. Therefore, C outputs D_i to A^I and adds D_i into the list L_3.

- If the list L_3 does not contain (ID_i, D_i, PK_i, x_i), C recovers the corresponding tuple (ID_i, t_i) from the list L_1, and sets $D_i = t_iP_{pub} = t_icP$, then outputs D_i to A^I. C also sets $PK_i = x_i =\perp$ and puts the tuple (ID_i, D_i, PK_i, x_i) into the list L_3.

  The probability of failure in partial private key extraction queries is at most $2/q_{H_1}$.

Public Key Queries: Suppose that A^I makes the query on an identity ID_i.

– If the list L_3 contains (ID_i, D_i, PK_i, x_i), C checks whether $PK_i =\perp$. If $PK_i \neq\perp$, C outputs PK_i to A^I. Otherwise, C selects a random value $r_i \in \mathbb{Z}_q^*$, and sets $PK_i = r_iP$ and $x_i = r_i$. C outputs PK_i to A^I and puts (PK_i, x_i) into the list L_3.

– If the list L_3 does not contain (ID_i, D_i, PK_i, x_i), C sets $D_i =\perp$, then it selects a random value $r_i \in \mathbb{Z}_q^*$, and sets $PK_i = r_iP$ and $x_i = r_i$. C outputs PK_i to A^I and writes the tuple (ID_i, D_i, PK_i, x_i) into the list L_3.

Private Key Extraction Queries: Suppose that A^I requests an identity ID_i. If $ID_i = ID_\eta$ or $ID_i = ID_\gamma$, then C fails and stops. If $ID_i \neq ID_\eta, ID_\gamma$, C looks up the list L_3 and runs the algorithm as follows.

– If the list L_3 contains (ID_i, D_i, PK_i, x_i), C checks whether $D_i =\perp$ and $PK_i =\perp$. If $D_i =\perp$, C makes a partial private key query itself to get D_i. If $PK_i =\perp$, C makes a public key query itself to obtain $PK_i = r_iP$, in which $x_i = r_i$. Then C adds these values into the list L_3 and outputs $SK_i = (D_i, x_i)$ to A^I.

– If the list L_3 does not contain (ID_i, D_i, PK_i, x_i), C makes a partial private key query and a public key query itself on ID_i, and then puts (ID_i, D_i, PK_i, x_i) into the list L_3 and outputs $SK_i = (D_i, x_i)$ to A^I.

Public Key Replacement Query: Suppose that A^I makes this query on (ID_i, PK_i').

– If the list L_3 contains the tuple (ID_i, D_i, PK_i, x_i), C sets $PK_i = PK_i'$ and $x_i = \perp$.

– If the list L_3 does not contain the tuple (ID_i, D_i, PK_i, x_i), C sets $D_i = \perp$, $PK_i = PK_i'$, and $x_i = \perp$. Then C saves the tuple (ID_i, D_i, PK_i, x_i) into the list L_3.

Deniable Authentication Queries: Suppose that A^I generates a message m and two identities ID_i and ID_j, C proceeds as follows.

– If $ID_i \neq ID_\eta, ID_\gamma$, C gets SK_i by running a private key extraction query and answers the query by running *Authenticate*(m, ID_i, PK_i, SK_i, ID_j, PK_j).

– If $ID_i = ID_\eta$ or $ID_i = ID_\gamma$, but $ID_j \neq ID_\eta, ID_\gamma$, C first randomly chooses $r \in \mathbb{Z}_q^*$ and computes $U = rQ_i$, and then C runs the H_2 simulation algorithm to get $h_2 = H_2(m, U, PK_i, PK_j, R)$ and computes $S = e(U + h_2Q_i, D_j)$ (C could get D_j from the partial private key query due to $ID_j \neq ID_\eta, ID_\gamma$). Finally, C sends $\sigma = (U, S)$ to A^I.

– If ID_i and ID_j are identities ID_η and ID_γ (i.e. $ID_i = ID_\eta$ and $ID_j = ID_\gamma$, or $ID_i = ID_\gamma$ and $ID_j = ID_\eta$), C first randomly selects r and h_2 from $\mathbb{Z}_q^*$ and sets $U = rP - h_2Q_i$ and $V = rP_{pub}$, and then C defines $H_2(m, U, PK_i, PK_j, R)$ as h_2 and adds the item $(m, U, PK_i, PK_j, R, h_2)$ into the list L_2. Finally, C computes $S = e(V, Q_{ID_j})$ and sends $\sigma = (U, S)$ to A^I. C fails if H_2 has been defined previously but this only happens with probability $(q_{da} + q_{H_2})/2^k$.

Verify Queries: Suppose that A^I makes the query with an input $\sigma = (U, S)$ for identities ID_i and ID_j.

– If $ID_j = ID_\eta, ID_\gamma$, then C fails and stops. The probability of failure in verify queries is at most $2/q_v$.

– If $ID_j \neq ID_\eta, ID_\gamma$, C gets SK_j by running a private key extraction query and answers the query by running *Verify*(σ, ID_i, PK_i, ID_j, PK_j, SK_j).

Eventually, A^I outputs a forgery quadruple $(m^*, \sigma^*, ID_a, ID_b)$, in which $\sigma^* = (U^*, S^*)$. We combine the identities $ID_c = \{ID_a, ID_b\}$ and the message m^* into a "generated " forged message (ID_c, m^*) so as to hide the identity-based aspect of the DA-CMA attacks, and simulate the setting of an identity-less adaptive-CMA existential forgery for which the forking lemma is proven. It follows from the forking lemma [21], if A^I is a sufficiently efficient forger in the above interaction, then we can construct a Las Vegas machine $A^{I'}$ that outputs two deniable authenticators $((ID_c, m^*), h_2^*, S^*)$ and $((ID_c, m^*), \bar{h_2^*}, \bar{S^*})$ with $h_2^* \neq \bar{h_2^*}$ and the same commitment U^*.

Finally, to solve the BDH problem given the machine $A^{I'}$ derived from A^I, we construct a machine C' as follows.

1. C' runs $A^{I'}$ to gain two distinct deniable authenticators $((ID_c, m^*), h_2^*, S^*)$ and $((ID_c, m^*), \bar{h_2^*}, \bar{S^*})$.

2. C' computes $e(P,P)^{abc}$ as $(S^*/\bar{S^*})^{1/(h_2^*-\bar{h_2^*})}$

Notice that the machine C' is our reduction from the BDH problem. Based on the forking lemma [21] and the lemma on the relationship between given-identity attack and chosen-identity attack [22]. If A^I succeeds in time t with probability $\epsilon \geq 5(q_{da}+1)(q_{da}+q_{H_2})q_{H_1}/(2^k-1)$, then C' can solve the BDH problem in expected time $t' \leq 60343 q_{H_2} q_{H_1} 2^k t/\epsilon(2^k-1)$. We should notice that the coefficient is changed because the simulator should select two different identities in advance.

Lemma 2. If a PPT adversary A^{II} has an advantage ϵ in forging a deniable authenticator in Game II, which runs in time t, and makes at most q_{H_i} queries to random oracle H_i for $i=1,2$, q_e queries to the private key extraction oracle, q_{pk} queries to the public key request oracle, q_{da} queries to the deniable authentication oracle, and q_v queries to the verify oracle. There exists a algorithm C can solve the CDH problem with probability
$\epsilon' > (\epsilon - (2/q_e + q_{da}(q_{da}+q_{H_2})+2)/2^k)$, within time $t' < t + (3q_{da}+q_v+2q_{H_2})t_e$ where t_e denotes the time required for one pairing evaluation.

Proof. Suppose that there is a Type II adversary A^{II} that can break our CL-DA protocol with the probability (t,ϵ). Then we can construct a algorithm C with advantage at least ϵ' within time at most t'. C receives a random instance (P, aP, bP) of the CDH problem and is required to compute abP. C will run A^{II} as a subroutine and act as A^{II}'s challenger in the DA-CMA game. C needs to maintain list L_1, L_2 which are initially empty to keep track values asked by A^{II} to random oracle queries H_1, H_2. Roughly speaking, these answers are randomly generated, whereas, to avoid collision and maintain consistency for answers to these hashing oracle, C keeps two lists L_1 and L_2 respectively to store the answers.

Without loss of generality, we assume that C will ask for $H_1(ID_i)$ before ID_i is used in private key extraction queries, public key queries, deniable authentication queries and verify queries. A^{II} never makes verify queries on a deniable authenticator, which is obtained from the deniable authentication queries. It just makes verify queries for observed deniable authenticators. Notice that both C and A^{II} can compute the partial private key $D_i = sH_1(ID_i)$, where s is the master private key.

C maintains a list $L_3 = (ID_i, PK_i, x_i)$ which does not need to be made in advance, and the list is populated when A^{II} makes certain queries as follows.

H_1 **Queries**: Suppose that A^{II} submits ID_i to oracle $H_1(\cdot)$. C first checks if the value of H_1 was previously defined. If it was, C returns the defined value. Otherwise, C chooses $r_i \in \mathbb{Z}_q^*$ randomly and sets $Q_i = r_i P$. Then it puts (ID_i, r_i) into the list L_1.

H_2 **Queries**: Suppose that A^{II} submits a tuple (m, U, PK_i, PK_j, R) to oracle $H_2(\cdot)$. C first checks whether H_2 has already been defined for that input. If so, C returns the existing value. Otherwise, C returns a random value $h_2 \in \mathbb{Z}_q^*$ as the answer. Then C puts the tuple (m, U, PK_i, PK_j, R) into the list L_2.

Public key Queries: Suppose that A^{II} makes the query on an identity ID_i.

- If the list L_3 contains (ID_i, PK_i, x_i), C returns PK_i to A^{II}.
- If the list L_3 does not contain (ID_i, PK_i, x_i), C selects a random value $r_i \in \mathbb{Z}_q^*$. At the η-th public key query, C answers by $PK_\eta = r_i aP$. At the γ-th public key query, C answers by $PK_\gamma = r_i bP$. For queries PK_i with $i \neq \eta, \gamma$, C answers by $PK_i = r_i P$ where $x_i = r_i$, and then puts (ID_i, PK_i, x_i) into the list L_3.

Private key Queries: Suppose that A^{II} makes the query on an identity ID_i.

- If the list L_3 contains the tuple (ID_i, PK_i, x_i), C returns $SK_i = (D_i, x_i)$ to A^{II}. In game II, D_i can be computed by C and A^{II}, so D_i is known.
- If the list L_3 does not contain the tuple (ID_i, PK_i, x_i), C makes a public key query on ID_i itself, and puts (ID_i, PK_i, x_i) into the list L_3. Then it outputs $SK_i = (D_i, x_i)$ to A^{II}.

Deniable Authentication Queries: Suppose that A^{II} generates a message m and two identities ID_i, ID_j, C proceeds as follows.

- If $ID_i \neq ID_\eta, ID_\gamma$, C gets SK_i by running a private key extraction query and answers the query by running *Authenticate*(m, ID_i, PK_i, SK_i, ID_j, PK_j).
- If $ID_i = ID_\eta$ or $ID_i = ID_\gamma$, but $ID_j \neq ID_\eta, ID_\gamma$, C randomly chooses $r \in \mathbb{Z}_q^*$ and computes $U = rQ_i$. Without loss of generality, we assume that the list L_3 contains a tuple

(ID_j, PK_j, x_j), and $PK_j \neq \perp$ (If the list L_3 does not contain such an entry, or $PK_j = \perp$, C runs a public key query to get (PK_j, x_j)). C runs the H_2 simulation algorithm to get $h_2 = H_2(m, U, PK_i, PK_j, R)$ and computes $S = e(U + h_2Q_i, D_j)$.

– If ID_i and ID_j are identities ID_η and ID_γ (i.e. $ID_i = ID_\eta$ and $ID_j = ID_\gamma$, or $ID_i = ID_\gamma$ and $ID_j = ID_\eta$), C randomly selects r and h_2 from $\mathbb{Z}_q^*$ and sets $U = rPK_i - h_2Q_i$ and $V = rsPK_i$. Then C defines $H_2(m, U, PK_i, PK_j, R)$ as h_2 and adds the item $(m, U, PK_i, PK_j, R, h_2)$ into the list L_2. Finally, C computes $S = e(V, Q_j)$ and sends $\sigma = (U, S)$ to A^{II}. C fails if H_2 has already been defined but this only happens with probability $(q_{da} + q_{H_2})/2^k$.

Verify Queries: Suppose that A^{II} queries the oracle with an input $\sigma = (U, S)$ for identities ID_i and ID_j.

– If $ID_j = ID_\eta, ID_\gamma$, C fails and stops. The probability of failure in verify queries is at most $2/q_v$.

– If $ID_j \neq ID_\eta, ID_\gamma$, without loss of generality, we assume that the list L_3 contains an item (ID_j, PK_j, x_j), and $PK_j = \perp$ (If the list L_3 does not contain such an entry, or if $PK_j = \perp$, C runs a public key query to get (PK_j, x_j)). C runs the H_2 simulation algorithm to look up the item $(m, U, PK_i, PK_j, R, h_2)$. It can obtain Q_j by calling H_1 queries. Then C computes $S = e(U + h_2Q_i, D_j)$.

Eventually, A^{II} outputs a valid deniable authenticator $\sigma^* = (U^*, S^*)$ from identity ID_η to identity ID_γ. It is easy to show that A^{II} will not realize that σ^* is not a valid deniable authenticator for the sender's private key SK_i and the receiver's Q_j unless it asks for the hash value $H_2(m, U, r_iaP, r_ibP, r_i^2abP)$. In this case, the solution of the CDH problem would be inserted in the list L_2. Then C looks up the list L_2 for tuples of the form (m, U, r_iaP, r_ibP, R). For each of them, C checks if $e(r_i^2P, R) = e(r_iaP, r_ibP)$. If the condition holds, C stops and outputs $R = abP$ as a solution of the CDH problem. If no such tuple satisfies the equality, C fails and stops.

Now we assess C's probability of failure. We saw that C fails if A^{II} asks the private key queries associated to ID_η or ID_γ with a probability exactly $2/q_e$. Also, the failure probability for C is at most $q_{da}(q_{da}+q_{H_2})/2^k$, since there is a conflict on H_2 in a deniable authentication query. The probability to reject a valid deniable authenticator is at most $2/2^k$. The bound on C's computation time derives from the fact that every deniable authentication query requires at most 3 pairing evaluations, every verify query requires one pairing evaluation. The extraction of the solution from L_2 implies to compute at most $2q_{H_2}$ pairings.

5.2 Performance

Table 1 shows a summary of comparing our protocol with other existing protocols [6, 12, 13, 14, 8, 15, 16, 9, 10, 17] in terms of security requirement and efficiency. We assume that these protocols [6, 8, 9, 10] are implemented on elliptic curve, which are based on PKI . For efficiency, suppose that $|G_1| = 160$ bits, $|G_2| = 1024$ bits, $|q| = 160$ bits, $|m| = 160$ bits, hash value = 160 bits and timestamp = 160bits. we denote by M the number of point multiplication operation in G_1, MM the number of multi-point multiplication operation in G_1 (which costs about 1.3 times more than single point multiplication) and P the number of pairing operation. The other operations are omitted because they are trivial. Among PKI-based protocols, the efficiency of [10] is the highest. Among ID-based protocols, [17] is the most efficient, since computation of the pairing is the most time-consuming. Our protocol is more efficient than [12, 13, 14, 15, 16], is same as to [17], and is lower than [6, 8, 9, 10]. Nevertheless, [6, 8, 9, 10] are based on PKI cryptography. The certificates management, such as generation, distribution, storage and revocation, are big problems. [12, 13, 14, 15, 16, 17] are based on ID-based cryptography which has the key escrow problems. For communication cost, since we need send an element S which belongs to G_2, our protocol is a little high. In addition, [6, 12, 13, 14] are interactive. Therefore, they have lower communication efficiency. For security, protocols [12, 13, 16] can not resist KCI attack.

6 Conclusion

In this paper, we first defined a security model for certificateless deniable authentication protocols, and then proposed an efficient non-interactive CL-DA protocol using bilinear pairing. Our protocol can be shown to be provably secure under the random oracle model, with the assumptions of the

Table 1: Comparison of efficiency and security for existing protocols

Protocols	Efficiency			Size	Non-interactive	Resist KCI attack	Type
	M	MM	P				
[6]	10.6	2	0	1140	N	Y	PKI
[8]	4	0	0	640	Y	Y	PKI
[9]	8.6	2	0	800	Y	Y	PKI
[10]	3	0	0	480	Y	Y	PKI
[12]	6	0	8	640	N	N	ID-based
[13]	8	0	8	800	N	N	ID-based
[14]	10	0	10	960	N	Y	ID-based
[15]	2	1	4	800	Y	Y	ID-based
[16]	4	0	2	480	Y	N	ID-based
[17]	3	0	2	1344	Y	Y	ID-based
Ours	3	0	2	1344	Y	Y	Certificateless

BDH problem and CDH problem.

References

[1] Aumann, Y., Rabin, M.: 'Authentication, enhanced security and error correcting codes'. Advances in Cryptology 1998, Santa Barbara, California, USA, August, 1998, pp. 299–303

[2] Dwork, C., Naor, M., Sahai, A.: 'Concurrent zero-knowledge'. Proc. Symposium on Theory of Computing-STOC 1998, Dallas, Texas, USA, May, 1998, pp. 409–418

[3] Aumann, Y., Rabin, M.: 'Efficient deniable authentication of long messages'. Webpage. http://www.cs.cityu.edu.hk/dept/video.html

[4] Deng, X., Lee, C. H., Zhu, H.: 'Deniable authentication protocols', IEE Proceedings-Computers and Digital Techniques, 2001, 148, (2), pp. 101–104

[5] Fan, L., Xu, C.X., Li, J. H.: 'Deniable authentication protocol based on Diffie–Hellman algorithm', Electronics Letters 2002, 38, (4), pp. 705–706

[6] Yoon, E. J., Ryu, E. K., Yoo, K. Y.: 'Improvement of Fan et al.'s Deniable Authentication Protocol based on Diffie Hellman Algorith', Applied Mathematics and Computation, 2005, 167, (1), pp. 274–280

[7] Tian, H., Chen, X., Wei, B., Liu, Y.: 'Security Analysis of a Suite of Deniable Authentication Protocols', International Journal of Network Security, 2013, 15, (6), pp. 369–374

[8] Shao, Z.: 'Efficient deniable authentication protocol based on generalized Elgamal signature scheme', Computer Standards and Interfaces, 2004, 26,(5), pp. 447–454

[9] Wang, B., Song, Z.: 'A non-interactive deniable authentication scheme based on designated verifier proofs', Information Sciences, 2009, 179, (6), pp. 858-865

[10] Tian, H., Chen, X., Jiang, Z.: 'Non-interactive deniable authentication protocols'. Proc. Information Security and Cryptology-Inscrypt 2011, Beijing, China, November, 2011, pp. 142–159

[11] Shamir, A.: 'Identity-based cryptosystems and signature schemes'. Advances in Cryptology 1984, Santa Barbara, California, USA, August, 1984, pp. 47–53

[12] Chou, J. S., Chen, Y., Huang, J. C.: 'A ID-Based Deniable Authentication Protocol on pairings', Cryptology ePrint Archive, Report, 2006, (335)

[13] Lim, M. H., Lee, S., Park, Y.: 'An enhanced ID-based deniable authentication protocol on pairings'. Proc. Computational Science and Its Applications-ICCSA 2007, Kuala Lumpur, Malaysia, August, 2007, pp. 1008–1017.

[14] Lim, M. H., Lee, S.: 'Cryptanalysis on improved Chou et al.'s ID-based deniable authentication protocol'. Proc. Information Systems Security-ICISS 2008, Hyderabad, Indian, December, 2008, pp. 87–93

[15] Shi, Y., Li, J.: 'Identity-based deniable authentication protocol', IEE Electronics Letter, 2005, 41, (5), pp. 241–242

[16] Cao, T., Lin, D., Xue, R.: 'An Efficient ID-based De-niable Authentication Protocol from Pairings'. Advanced Information Networking and Applications-AINA 2005, IEEE Computer Society, Taipei, Taiwan, March, 2005, pp. 388–391

[17] Li, F., Xiong, P., Jin, C.: 'Identity-Based Deniable Authentication for Ad Hoc Networks', Computing, 2013, doi:10.1007/s00607-013-0321-5

[18] Al-Riyami, S., Paterson, K.: 'Certificateless public key cryptography'. Advances in Cryptology-Asiacrypt 2003, Taipei, Taiwan, November, 2003, pp. 452–473

[19] Zhang, Z., Wong, D., Xu J., and Feng, D.: 'Certificateless Public-Key Signature: Security Model and Efficient Construction'. Proc. Applied Cryptography and Network Security-ACNS 2006, Singapore, June, 2006, pp. 293–308

[20] Boneh, D., Franklin, M.: 'Identity-based encryption from the weil pairing', SIAM Journal on Computing, 2003, 32, (3), pp. 586–615.

[21] Pointcheval, D., Stern, J.: 'Security arguments for digital signatures and blind signatures', Journal Cryptography, 2000, 13, (3), pp. 361–396

[22] Cha, J. C., Cheon, J. H.: 'An identity-based signature from gap Diffie-Hellman groups'. Proc. Public Key Cryptography-PKC 2003, Miami, Florida, USA, January, 2003, pp. 18–30