Telecommun Syst (2016) 62:625–639
DOI 10.1007/s11235-015-0099-1

Identity-based deniable authenticated encryption and its application to e-mail system

Fagen Li[1] · **Zhaohui Zheng**[1] · **Chunhua Jin**[1]

Published online: 22 October 2015

Abstract An authenticated encryption (AE) scheme simultaneously achieves two security goals: confidentiality and authenticity. AE can be divided into symmetric AE and asymmetrical (public key) AE. In a symmetric AE scheme, deniability is gained automatically. However, a public key AE scheme can not gain deniability automatically; on the contrary, it provides non-repudiation. In this paper, we address a question on deniability of public key AE. Of course, we can achieve this goal by "deniable authentication followed by encryption" method. However, such method has the following two weaknesses: (1) the computational cost and communication overhead are the sum of two cryptographic primitives; (2) it is complex to design cryptographic protocols with deniable authentication and confidentiality using two cryptographic primitives. To overcome the two weaknesses, we propose a new concept called deniable authenticated encryption (DAE) that can achieve both the functions of deniable authentication and public key encryption simultaneously, at a cost significantly lower than that required by the "deniable authentication followed by encryption" method. This single cryptographic primitive can simplify the design of cryptographic protocols with deniable authentication and confidentiality. In particular, we construct an identity-based deniable authenticated encryption (IBDAE) scheme. Our construction uses tag-key encapsulation mechanism (KEM) and data encapsulation mechanism (DEM) hybrid techniques, which is more practical for true applications. We show how to construct an IBDAE scheme using an identity-based deniable authenticated tag-KEM (IBDATK) and a DEM. We also propose an IBDATK scheme and prove its security in the random oracle model. For typical security level, our scheme is at least 50.7 and 22.7 % faster than two straightforward "deniable authentication followed by encryption" schemes, respectively. The communication overhead is respectively reduced at least 21.3 and 31.1 %. An application of IBDAE to an e-mail system is described.

Keywords Security · Authenticated encryption · Identity-based cryptography · Deniability · E-mail system

1 Introduction

An authenticated encryption (AE) scheme simultaneously achieves two security goals: confidentiality and authenticity. AE can be divided into symmetric AE [1] and asymmetrical (public key) AE [2,3]. A symmetric AE use a keyed hash (i.e, a MAC) with some appropriate key K_1 along with a secure encryption scheme with an independent key K_2 to achieve AE. In this model, we need agree K_1 and K_2 between the sender and the receiver in advance. The authenticity of symmetric AE is deniable authentication since both the sender and the receiver can generate the same ciphertext. That is, the receiver can generate a ciphertext that is indistinguishable from that generated by the sender. A public key AE integrates public key encryption and digital signature in a single procedure to reduce the computation and communication cost. A public key AE does not automatically achieve deniable authentication since only the sender can generate a valid ciphertext. That is, a public key AE scheme provides non-repudiation. Therefore, there is an important difference about authenticity between symmetric AE and public key AE. The symmetric AE is deniable and the public key AE is not deniable.

✉ Fagen Li
fagenli@uestc.edu.cn

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

1.1 Motivation and contribution

In this paper, we address a question on deniability of public key AE. Of course, we can achieve this goal by "deniable authentication followed by encryption" method. However, such method has the following two weaknesses: (1) the computational cost and communication overhead are the sum of two cryptographic primitives; (2) it is complex to design cryptographic protocols with deniable authentication and confidentiality using two cryptographic primitives. In this paper, we propose a new concept called deniable authenticated encryption (DAE) that can achieve both the functions of deniable authentication and public key encryption simultaneously, at a cost significantly lower than that required by the "deniable authentication followed by encryption" method. This single cryptographic primitive can simplify the design of cryptographic protocols with deniable authentication and confidentiality. In particular, we construct an identity-based deniable authenticated encryption (IBDAE) scheme. Our construction uses tag-key encapsulation mechanism (KEM) and data encapsulation mechanism (DEM) hybrid techniques, which is more practical for true applications. We show how to construct an IBDAE scheme using an identity-based deniable authenticated tag-KEM (IBDATK) and a DEM. We also propose an IBDATK scheme and prove its security in the random oracle model. An application of IBDAE to an e-mail system is described.

1.2 Related work

Here we introduce four related notions, identity-based cryptography (IBC), hybrid encryption, and deniable authentication.

IBC is introduced by Shamir [4] in 1984. Compared with the public key infrastructure (PKI), the main advantage of the IBC is the elimination of public key certificates. In the IBC, a user's public key can be derived directly from its identity information, such as telephone numbers, e-mail addresses and IP addresses. The private key of the user is generated by a trusted third party called private key generator (PKG). Authenticity of the public key is explicitly verified without requiring an attached public key certificate. The IBC is very suitable for developing a secure e-mail system. When Alice hopes to send an e-mail to Bob at `bob@uestc.edu.cn`, she encrypts her message using the string "`bob@uestc.edu.cn`". In this process, Alice does not need to obtain Bob's public key certificate. When Bob receives the encrypted e-mail, he applies for a private key from the PKG and then decrypts his e-mail. Note that unlike the PKI-based e-mail systems, Alice can send an encrypted e-mail to Bob even if Bob has not obtained his key pair information. In 2001, Boneh and Franklin [5] designed a practical identity-based encryption (IBE) scheme using bilinear pairings and proved its security in the random oracle model. Park and Lee [6] proposed another IBE scheme that achieves a tight security reduction to the decisional bilinear Diffie–Hellman assumption in the random oracle model. In 2002, Hess [7] constructed an identity-based signature (IBS) whose security depends on the hardness of the Diffie–Hellman problem in the random oracle model. In 2003, Cha and Cheon [8] designed a new IBS using gap Diffie–Hellman groups. In 2011, Hsu and Lin [9,10] extended AE into the identity-based environment and constructed identity-based authenticated encryption (IBAE). However, these two IBAE schemes have non-repudiation. That is, they are not deniable. Some identity-based signcryption (IBSC) [11–13] also were proposed to achieve both confidentiality and authentication. However, these IBSC schemes still have non-repudiation.

The practical way to perform secret communication for large messages is to use hybrid encryption. The hybrid encryption splits the encryption process into two parts: one part uses public key techniques to encrypt a one-time symmetric key; the other part uses the symmetric key to encrypt the actual message. In such a construction, the public key part is called key encapsulation mechanism (KEM) and the symmetric key part is called data encapsulation mechanism (DEM). In 2003, Cramer and Shoup [14] first gave a formal analysis for the hybrid KEM–DEM construction. Then some efficient KEM–DEM constructions have been proposed [15–17]. The advantage of this paradigm is that it gives a clear separation between the various parts of the cipher allowing for modular design. In [15], Abe et al. introduced tag-KEM that takes as input a tag in KEM. Note that the using of tag-KEM brings simpler scheme descriptions and better generic security reductions. Bentahar et al.'s [18] extended KEM into identity-based environment and proposed several efficient constructions of identity-based KEM. In 2014, Abdalla et al. [19] discussed the relation between the notions of verifiable random functions and identity-based KEM.

Deniable authentication is different to traditional authentication and has the following two main characteristics: (1) it enables an intended receiver to identify the source of a given message; (2) the intended receiver cannot prove the source of a given message to any third party. Deniable authentication can be used in many specialized applications. For example, it can provide freedom from coercion in electronic voting systems and secure negotiation over the Internet [20,21]. In 2005, Shi and Li [22] extended deniable authentication into identity-based environment and designed an identity-based deniable authentication (IBDA) protocol. Kar [23] designed an IBDA protocol based on Diffie–Hellman problem assumption. However, Kar's protocol is interactive. An interactive protocol usually consumes more communication cost than a non-interactive protocol. In 2014, Li et al. [24] proposed a non-interactive IBDA protocol using bilinear pairings under the bilinear Diffie–Hellman assumption. In addition, their

protocol supports batch verification that can speed up the verification of authenticators.

1.3 Organization

The rest of this paper is organized as follows. We define the formal model of IBDAE in Sect. 2. The IBDATK is given in Sect. 3. We show how to construct an IBDAE scheme using an IBDATK and a DEM in Sect. 4. An example of IBDATK is described in Sect. 5. We discuss the performance of our scheme in Sect. 6. A secure e-mail protocol based on IBDAE is designed in Sect. 7. Finally, the conclusions are given in Sect. 8.

2 IBDAE

In this section, we give the formal definition and security notions for IBDAE.

2.1 Syntax

A generic IBDAE scheme consists of the following four algorithms.

- `Setup` is a probabilistic algorithm run by a PKG that takes as input a security parameter k, and outputs a master secret key s and the system parameters *param* including a master public key P_{pub}. Here we assume that *param* are public so that we do not need to include them in other algorithms.
- `Extract` is a key extraction algorithm run by the PKG that takes as input an identity ID and the master secret key s, and outputs the corresponding private key S_{ID}. The PKG transmits the private key to its owner in a secure way.
- `Encrypt` is a probabilistic deniable authenticated encryption algorithm run by a sender that takes as input a message m, a sender's private key S_{ID_s} and a receiver's identity ID_r, and outputs a ciphertext σ.
- `Decrypt` is a deterministic deniable authenticated decryption algorithm run by a receiver that takes as input a ciphertext σ, a sender's identity ID_s and a receiver's private key S_{ID_r}, and outputs a plaintext m or the symbol $\perp$ if σ is an invalid ciphertext between the sender and the receiver.

For consistency, we require that if $\sigma = \texttt{Encrypt}(m, S_{ID_s}, ID_r)$, then we have $m = \texttt{Decrypt}(\sigma, ID_s, S_{ID_r})$. Note that here `Encrypt` and `Decrypt` are different from common Encrypt and Decrypt in public key encryption schemes. Here `Encrypt` means deniable authenticated encryption and `Decrypt` means deniable authenticated decryption. In the following contents, encryption and decryption usual mean deniable authenticated encryption and deniable authenticated decryption, respectively.

2.2 Security notions

An IBDAE scheme should satisfy confidentiality and deniable authentication. Here we give two games to capture the two security properties.

The standard adopted security notion for confidentiality is the indistinguishability against adaptive chosen ciphertext attack (IND-CCA2). We apply this notion to our IBDAE. We consider the following game (Game-I) played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- `Initial` $\mathcal{C}$ runs `Setup` algorithm with a security parameter k and sends the system parameters *param* to $\mathcal{A}$.
- `Phase 1` $\mathcal{A}$ performs a polynomially bounded number of queries (these queries may be made adaptively, i.e. each query may depend on the answer to the previous queries).
  - *Key extraction queries* $\mathcal{A}$ chooses an identity ID. $\mathcal{C}$ runs `Extract` algorithm and sends the corresponding private key S_{ID} to $\mathcal{A}$.
  - *Encryption queries* $\mathcal{A}$ produces a plaintext m and two identities ID_i and ID_j. $\mathcal{C}$ first runs `Extract` algorithm to generate the sender's private key S_{ID_i}. Then $\mathcal{C}$ runs $\texttt{Encrypt}(m, S_{ID_i}, ID_j)$ algorithm and sends σ to $\mathcal{A}$.
  - *Decryption queries* $\mathcal{A}$ produces a ciphertext σ and two identities ID_i and ID_j. $\mathcal{C}$ first runs `Extract` algorithm to generate the receiver's private key S_{ID_j}. Then $\mathcal{C}$ runs $\texttt{Decrypt}(\sigma, ID_i, S_{ID_j})$ algorithm and sends the result to $\mathcal{A}$ (this result can be the $\perp$ symbol if σ is an invalid ciphertext).
- `Challenge` $\mathcal{A}$ decides when phase 1 ends. $\mathcal{A}$ generates two equal length plaintexts m_0 and m_1 and two identities ID_s and ID_r on which it wants to be challenged. It can not have asked the private key corresponding to ID_r in the phase 1. $\mathcal{C}$ takes a random bit $\beta \in \{0, 1\}$ and computes $\sigma^* = \texttt{Encrypt}(m_\beta, S_{ID_s}, ID_r)$ which is sent to $\mathcal{A}$.
- `Phase 2` $\mathcal{A}$ can ask a polynomially bounded number of queries adaptively again as in the phase 1. This time, it can not ask a key extraction query on ID_r and can not ask a decryption query on (σ^*, ID_s, ID_r) to obtain the corresponding plaintext.
- `Guess` $\mathcal{A}$ produces a bit β' and wins the game if $\beta' = \beta$.

The advantage of $\mathcal{A}$ is defined as $\text{Adv}(\mathcal{A}) = |2\Pr[\beta' = \beta] - 1|$, where $\Pr[\beta' = \beta]$ denotes the probability that $\beta' = \beta$.

Definition 1 An adversary $\mathcal{A}$ is said to be an $(\epsilon_{dae}, t, q_k, q_e, q_d)$-attacker of an IBDAE scheme if $\mathcal{A}$ has advantage at least ϵ_{dae} in the Game-I, runs in time at most t, and makes at most q_k key extraction queries, q_e encryption queries and q_d decryption queries. An IBDAE scheme is said to be $(\epsilon_{dae}, t, q_k, q_e, q_d)$-IND-CCA2 secure if no $(\epsilon_{dae}, t, q_k, q_e, q_d)$-attacker exists.

Note that $\mathcal{A}$ is allowed to make a key extraction query on identity ID_s in the Game-I. It ensures the forward security of the scheme, i.e. confidentiality is preserved in case the sender's private key becomes compromised.

For the deniable authentication, we adopt deniable authentication against adaptive chosen messages attack (DA-CMA) notion in [24]. We consider the following game (Game-II) played between a challenger $\mathcal{C}$ and an adversary $\mathcal{F}$.

- `Initial` $\mathcal{C}$ runs `Setup` algorithm with a security parameter k and sends the system parameters *param* to $\mathcal{F}$.
- `Attack` $\mathcal{F}$ performs a polynomially bounded number of queries just like in the Game-I.
- `Forgery` $\mathcal{F}$ produces a triple (σ^*, ID_s, ID_r) and wins the game if the following conditions hold:
  1. $\texttt{Decrypt}(\sigma^*, ID_s, S_{ID_r}) = m^*$.
  2. $\mathcal{F}$ has not asked key extraction queries on ID_s and ID_r.
  3. $\mathcal{F}$ has not asked an encryption query on (m^*, ID_s, ID'_r). Here ID'_r may be different from ID_r.

The advantage of $\mathcal{F}$ is defined as the probability that it wins.

Definition 2 An adversary $\mathcal{F}$ is said to be an $(\epsilon_{dae}, t, q_k, q_e, q_d)$-forger of an IBDAE scheme if $\mathcal{F}$ has advantage at least ϵ_{dae} in the Game-II, runs in time at most t, and makes at most q_k key extraction queries, q_e encryption queries and q_d decryption queries. An IBDAE scheme is said to be $(\epsilon_{dae}, t, q_k, q_e, q_d)$-DA-CMA secure if no $(\epsilon_{dae}, t, q_k, q_e, q_d)$-forger exists.

Note that $\mathcal{F}$ is not allowed to make a key extraction query on identity ID_r in the Game-II. This condition is necessary to obtain the deniability property. The sender can deny its action since the receiver also can generate a valid ciphertext. This is the main difference between deniable authentication in DAE and undeniable authentication in digital signature.

3 IBDATK

In this section, we give the formal definition and security notions for identity-based deniable authenticated tag-KEM (IBDATK). IBDATK can be considered as an identity-based tag-KEM with deniable authentication property. IBDATK should provide two important security properties: confidentiality and deniable authentication.

3.1 Syntax

A generic IBDATK scheme consists of the following five algorithms.

- `Setup` is a probabilistic algorithm run by a PKG that takes as input a security parameter k, and outputs the system's parameters *param*, including a master public key P_{pub} and a master secret key s. We also assume that *param* are public so that we do not need to include them in other algorithms.
- `Extract` is a key extraction algorithm run by the PKG that takes as input an identity ID and the master secret key s, and outputs the corresponding private key S_{ID}. The PKG transmits the private key to its owner in a secure way.
- `Sym` is a probabilistic symmetric key generation algorithm run by a sender that takes as input a sender's private key S_{ID_s} and a receiver's identity ID_r, and outputs a symmetric key K together with internal state information ω. Here $K \in \mathcal{K}_{\text{IBDATK}}$ is a key in the space of possible session keys at a given security level. $\mathcal{K}_{\text{IBDATK}}$ is the key space.
- `Encap` is a probabilistic key encapsulation algorithm which takes as input the state information ω and an arbitrary tag τ, and returns an encapsulation $\psi \in \mathcal{E}_{\text{IBDATK}}$. Here $\mathcal{E}_{\text{IBDATK}}$ is the encapsulation space.
- `Decap` is a deterministic decapsulation algorithm run by a receiver that takes as input an encapsulation ψ, a tag τ, a sender's identity ID_s and a receiver's private key S_{ID_r}, and outputs the corresponding key K or a special symbol $\perp$ indicating invalid encapsulation.

We make the consistency constraint that if $(K, \omega) = \texttt{Sym}(S_{ID_s}, ID_r)$ and $\psi = \texttt{Encap}(\omega, \tau)$, then $K = \texttt{Decap}(\psi, \tau, ID_s, S_{ID_r})$.

3.2 Security notions

An IBDATK scheme should satisfy confidentiality and deniable authentication. We give the formal security definition here. For the confidentiality, we consider the following game (Game-III) played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- `Initial` $\mathcal{C}$ runs `Setup` algorithm with a security parameter k and sends the system parameters *param* to $\mathcal{A}$.
- `Phase 1` $\mathcal{A}$ performs a polynomially bounded number of queries (these queries may be made adaptively, i.e.

each query may depend on the answer to the previous queries).

- *Key extraction queries* $\mathcal{A}$ chooses an identity ID. $\mathcal{C}$ runs Extract algorithm and sends the corresponding private key S_{ID} to $\mathcal{A}$.
- *Symmetric key generation queries* $\mathcal{A}$ produces a sender's identity ID_i and a receiver's identity ID_j. $\mathcal{C}$ first runs Extract algorithm to generate the sender's private key S_{ID_i} and runs $(K, \omega) = \mathtt{Sym}(S_{ID_i}, ID_j)$. It then stores the value ω (hidden from the view of the adversary, and overwriting any previously stored values), and sends the symmetric key K to $\mathcal{A}$.
- *Key encapsulation queries* $\mathcal{A}$ produces an arbitrary tag τ. $\mathcal{C}$ checks whether there exists a stored value ω. If not, it returns $\perp$ and terminates. Otherwise it erases the value from storage and returns $\psi = \mathtt{Encap}(\omega, \tau)$ to $\mathcal{A}$.
- *Key decapsulation queries* $\mathcal{A}$ chooses an encapsulation ψ, a tag τ, a sender's identity ID_i and a receiver's identity ID_j. $\mathcal{C}$ first runs Extract algorithm to generate the receiver's private key S_{ID_j}. Then $\mathcal{C}$ runs $\mathtt{Decap}(\psi, \tau, ID_i, S_{ID_j})$ algorithm and sends the result to $\mathcal{A}$.

- Challenge $\mathcal{A}$ decides when phase 1 ends. $\mathcal{A}$ generates a sender's identity ID_s and a receiver's identity ID_r on which it wishes to be challenged. The identity ID_r should not appear in any key extraction queries in phase 1. $\mathcal{C}$ first runs $(K_1, \omega^*) = \mathtt{Sym}(S_{ID_s}, ID_r)$. Then $\mathcal{C}$ chooses $K_0 \in \mathcal{K}_{\mathrm{IBDATK}}$ and a bit $\beta \in \{0, 1\}$ randomly, and sends K_β to $\mathcal{A}$. When $\mathcal{A}$ receives K_β, it may ask the same queries as previously. Then $\mathcal{A}$ generates a tag τ^*. Finally, $\mathcal{C}$ computes $\psi^* = \mathtt{Encap}(\omega^*, \tau^*)$ and sends it to $\mathcal{A}$ as a challenge encapsulation.
- Phase 2 $\mathcal{A}$ can ask a polynomially bounded number of queries adaptively again as in phase 1 with the restriction that it can not ask a key extraction query on ID_r and can not ask a decapsulation query on $(\psi^*, \tau^*, ID_s, ID_r)$ to obtain the corresponding key.
- Guess $\mathcal{A}$ produces a bit β' and wins the game if $\beta' = \beta$.

The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}(\mathcal{A}) = |2\Pr[\beta' = \beta] - 1|$, where $\Pr[\beta' = \beta]$ denotes the probability that $\beta' = \beta$.

Definition 3 An adversary $\mathcal{A}$ is said to be an $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$-attacker of an IBDATK scheme if $\mathcal{A}$ has advantage at least ϵ_{datk} in the Game-III, runs in time at most t, and asks at most q_k key extraction queries, q_s symmetric key generation queries, q_e key encapsulation queries and q_d key decapsulation queries. An IBDATK scheme is said to be $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$-IND-CCA2 secure if no $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$-attacker exists.

For deniable authentication, we consider the following game (Game-IV) played between a challenger $\mathcal{C}$ and an adversary $\mathcal{F}$.

- Initial $\mathcal{C}$ runs Setup algorithm with a security parameter k and sends the system parameters *param* to $\mathcal{F}$.
- Attack $\mathcal{F}$ performs a polynomially bounded number of queries just like in the Game-III.
- Forgery $\mathcal{F}$ produces a quaternion $(\psi^*, \tau^*, ID_s, ID_r)$ and wins the game if the following conditions hold:
  1. $\mathtt{Decap}(\psi^*, \tau^*, ID_s, S_{ID_r}) = K^*$.
  2. $\mathcal{F}$ has not asked key extraction queries on ID_s and ID_r.
  3. $\mathcal{F}$ has not asked a key encapsulation query on τ^*.

The advantage of $\mathcal{F}$ is defined as the probability that it wins.

Definition 4 An adversary $\mathcal{F}$ is said to be an $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$-forger of an IBDATK scheme if $\mathcal{F}$ has advantage at least ϵ_{datk} in the Game-IV, runs in time at most t, and asks at most q_k key extraction queries, q_s symmetric key generation queries, q_e key encapsulation queries and q_d key decapsulation queries. An IBDATK scheme is said to be $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$-DA-CMA secure if no $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$-forger exists.

4 A hybrid IBDAE scheme

We can combine an IBDATK scheme IBDATK with a DEM scheme DEM (the definition of DEM can be found in Appendix 1) to form a hybrid IBDAE scheme IBDAE. Our method is described in Fig. 1. Note that the tag is the ciphertext output by the DEM. Such construction yields simpler scheme descriptions and better generic security reductions.

We give the security results for such construction in Theorems 1 and 2.

Theorem 1 *Let* IBDAE *be a hybrid IBDAE scheme constructed from an IBDATK scheme* IBDATK *and a DEM scheme* DEM. *If the* IBDATK *is* $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$*-IND-CCA2 secure and the* DEM *is* $(\epsilon_{dem}, \bar{t})$*-IND-PA secure, then the* IBDAE *is* $(\epsilon_{dae}, t', q'_k, q'_e, q'_d)$*-IND-CCA2 secure, where* $\epsilon_{datk} \geq (\epsilon_{dae} - \epsilon_{dem})/2$, $t = t' + \bar{t} + O(q'_e T_{enc} + q'_d T_{dec})$, $q_k = q'_k$, $q_s = q_e = q'_e$, $q_d = q'_d$. *Here* T_{enc} *and* T_{dec} *are the maximum time for computing an encryption in* DEM *and a decryption in* DEM.

Proof See Appendix 2. □

Theorem 2 *Let* IBDAE *be a hybrid IBDAE scheme constructed from an IBDATK scheme* IBDATK *and a DEM scheme* DEM. *If the* IBDATK *is* $(\epsilon_{datk}, t, q_k, q_s, q_e, q_d)$*-DA-CMA secure, then the* IBDAE *is* $(\epsilon_{dae}, t', q'_k, q'_e, q'_d)$*-DA-CMA secure, where* $\epsilon_{datk} \geq \epsilon_{dae}$, $t = t' + O(q'_e T_{enc} +$

Fig. 1 Construction of IBDAE from IBDATK and DEM

`IBDAE.Setup:` On input a security parameter k:
1. $(param, s) = \texttt{IBDATK.Setup}(k)$
2. Output the system parameters $param$ and the master secret key s

`IBDAE.Extract:` On input an identity $ID \in \{0,1\}^*$ and the master secret key s:
1. $S_{ID} = \texttt{IBDATK.Extract}(ID, s)$
2. Output the private key S_{ID} for the identity ID

`IBDAE.Encrypt:` On input a message $m \in \{0,1\}^*$, a sender's private key S_{ID_s} and a receiver's identity ID_r:
1. $(K, \omega) = \texttt{IBDATK.Sym}(S_{ID_s}, ID_r)$
2. $c = \texttt{DEM.Enc}(K, m)$
3. $\psi = \texttt{IBDATK.Encap}(\omega, c)$
4. Output the ciphertext $\sigma = (\psi, c)$

`IBDAE.Decrypt:` On input a ciphertext σ, a sender's identity ID_s and a receiver private key S_{ID_r}:
1. $K = \texttt{IBDATK.Decap}(\psi, c, ID_s, S_{ID_r})$
2. If $K = \perp$, then output $\perp$ and stop
3. $m = \texttt{DEM.Dec}(K, c)$
4. Output the message m

$q'_d T_{dec})$, $q_k = q'_k$, $q_s = q_e = q'_e$, $q_d = q'_d$. *Here* T_{enc} *and* T_{dec} *are the maximum time for computing an encryption in* DEM *and a decryption in* DEM.

Proof See Appendix 3. □

5 An efficient IBDATK scheme

In this section, we propose an efficient IBDATK scheme using bilinear pairings. We first describe the basic definition and properties of the bilinear pairings. The we give the IBDATK scheme and discuss its consistency, deniability and security.

5.1 Bilinear pairings

Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. A bilinear pairing is a map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with the following properties:

1. *Bilinearity* $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$, $a, b \in \mathbb{Z}_q^*$.
2. *Non-degeneracy* There are P and $Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$, where 1 is the identity element of group G_2.
3. *Computability* There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G_1$.

The modified Weil pairing and the Tate pairing [5,7,8] are admissible maps of this kind. The security of our scheme described here relies on the hardness of the following problems.

Given two groups G_1 and G_2 of the same prime order q, a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ and a generator P of G_1, the decisional bilinear Diffie–Hellman (DBDH) problem in $(G_1, G_2, \hat{e})$ is to decide whether $\theta = \hat{e}(P, P)^{abc}$ or not given (P, aP, bP, cP) and an element $\theta \in G_2$. We define the advantage of an adversary $\mathcal{C}$ against the DBDH like this

$$\begin{aligned}\mathrm{Adv}(\mathcal{C}) = |&P_{a,b,c,\in\mathbb{Z}_q^*,\theta\in G_2}[\mathcal{C}(P, aP, bP, cP, \theta) = 1] \\ &- P_{a,b,c,\in\mathbb{Z}_q^*}[\mathcal{C}(P, aP, bP, cP, \hat{e}(P, P)^{abc}) = 1]|.\end{aligned}$$

Definition 5 The (ϵ_{dbdh}, t)-DBDH assumption holds if no t-polynomial time adversary $\mathcal{C}$ has advantage at least ϵ_{dbdh} in solving the DBDH problem.

Given two groups G_1 and G_2 of the same prime order q, a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ and a generator P of G_1, the bilinear Diffie–Hellman (BDH) problem in $(G_1, G_2, \hat{e})$ is to compute $h = \hat{e}(P, P)^{abc}$ given (P, aP, bP, cP).

Definition 6 The (ϵ_{bdh}, t)-BDH assumption holds if no t-polynomial time adversary $\mathcal{C}$ has advantage at least ϵ_{bdh} in solving the BDH problem.

5.2 Our scheme

Our scheme consists of the following five algorithms.

– `Setup` Define G_1, G_2 and $\hat{e}$ as in Sect. 5.1. Let H_1, H_2 and H_3 be three hash functions where $H_1 : \{0,1\}^* \rightarrow$

G_1, $H_2 : G_2 \to \{0,1\}^n$ and $H_3 : \{0,1\}^* \times G_2 \to \mathbb{Z}_q^*$. Here n is the key length of a DEM and $q \geq 2^k$, where k is a security parameter. Let P be a generator of G_1. The PKG chooses a master secret key $s \in \mathbb{Z}_q^*$ randomly and computes $P_{pub} = sP$. The PKG publishes system parameters $param = \{G_1, G_2, q, n, \hat{e}, P, P_{pub}, H_1, H_2, H_3\}$ and keeps the master secret key s secret.

- `Extract` Given an identity ID, the PKG computes the corresponding private key $S_{ID} = sQ_{ID}$ and sends it to its owner in a secure way. Here $Q_{ID} = H_1(ID)$.
- `Sym` Given a sender's private key S_{ID_s} and a receiver's identity ID_r, this algorithm works as follows.
  1. Choose x from $\mathbb{Z}_q^*$ randomly.
  2. Compute $z = \hat{e}(P_{pub}, Q_{ID_r})^x$.
  3. Output $K = H_2(z)$ and $\omega = (x, z, S_{ID_s}, ID_s, ID_r)$.
- `Encap` Given the state information ω and an arbitrary tag τ, this algorithm works as follows.
  1. Compute $u = H_3(\tau, z)$.
  2. Compute $V = uS_{ID_s} + xP_{pub}$ and $T = \hat{e}(V, Q_{ID_r})$.
  3. Compute $R = uQ_{ID_s}$.
  4. Output $\psi = (R, T)$.
- `Decap` Given an encapsulation $\psi = (R, T)$, a tag τ, a sender's identity ID_s and a receiver's private key S_{ID_r}, this algorithm works as follows.
  1. Compute $z = T/\hat{e}(R, S_{ID_r})$.
  2. Compute $u = H_3(\tau, z)$.
  3. If $R = uQ_{ID_s}$, output the $K = H_2(z)$, otherwise output the symbol $\perp$.

5.3 Consistency

The consistency of our scheme can be easily verified by the following equations.

$$
\begin{aligned}
z &= \frac{T}{\hat{e}(R, S_{ID_r})} = \frac{\hat{e}(V, Q_{ID_r})}{\hat{e}(R, S_{ID_r})} = \frac{\hat{e}(uS_{ID_s} + xP_{pub}, Q_{ID_r})}{\hat{e}(R, S_{ID_r})} \\
&= \frac{\hat{e}(uS_{ID_s}, Q_{ID_r})\hat{e}(xP_{pub}, Q_{ID_r})}{\hat{e}(R, S_{ID_r})} \\
&= \frac{\hat{e}(uQ_{ID_s}, S_{ID_r})\hat{e}(P_{pub}, Q_{ID_r})^x}{\hat{e}(R, S_{ID_r})} \\
&= (P_{pub}, Q_{ID_r})^x.
\end{aligned}
$$

5.4 Deniability

The receiver with private key S_{ID_r} can generate an encapsulation that is indistinguishable from that generated by the sender with private key S_{ID_s}. To simulate the transcripts on a tag τ, the receiver performs the steps below.

1. Choose $\bar{x}$ from $\mathbb{Z}_q^*$ randomly.
2. Compute $\bar{z} = \hat{e}(P_{pub}, Q_{ID_r})^{\bar{x}}$.
3. Compute $\bar{u} = H_3(\tau, \bar{z})$.
4. Compute $\bar{R} = \bar{u}Q_{ID_s}$.
5. Compute $\bar{T} = \bar{z} \cdot \hat{e}(\bar{R}, S_{ID_r})$.

$\bar{\psi} = (\bar{R}, \bar{T})$ generated by the receiver is indistinguishable from $\psi = (R, T)$ that is generated by the sender according to the `Encap` algorithm in Sect. 5.2. Let $\psi' = (R', T')$ be an encapsulation that is randomly chosen in the set of all valid sender's encapsulation intended to receiver. The probability $\Pr[\bar{\psi} = \psi']$ is $1/(q-1)$ because $\bar{\psi}$ is generated from a randomly chosen value $\bar{x} \in \mathbb{Z}_q^*$. Likewise, the probability $\Pr[\psi = \psi']$ has the same value $1/(q-1)$ because it is generated from $x \in \mathbb{Z}_q^*$. That is, both distributions are the same.

5.5 Security

Theorem 3 *In the random oracle model, if an adversary $\mathcal{A}$ has a non-negligible advantage ϵ_{datk} against the IND-CCA2 security of the proposed scheme when running in a time t and performing q_k key extraction queries, q_s symmetric key generation queries, q_e key encapsulation queries, q_d key decapsulation queries and q_{H_i} queries to oracles H_i ($i = 1, 2, 3$), then there exists an algorithm $\mathcal{C}$ that can solve the DBDH problem with an advantage*

$$\epsilon_{dbdh} \geq \frac{\epsilon - q_d/2^{k-1}}{2q_{H_1}}$$

in a time $t' \leq t + O(q_s + q_e + q_d)t_p$, where t_p denotes the cost for one pairing operation.

Proof See Appendix 4. □

Theorem 4 *In the random oracle model, if an adversary $\mathcal{F}$ is able to win the Game-IV with an advantage $\epsilon \geq 5(q_e+1)(q_e+q_{H_3})q_{H_1}/(2^k-1)$ within a time t for a security parameter k and asking at most q_k key extraction queries, q_s symmetric key generation queries, q_e key encapsulation queries, q_d key decapsulation queries and q_{H_i} queries to oracles $H_i (i = 1, 2, 3)$, then there exists an algorithm $\mathcal{C}'$ that can solve the BDH problem in expected time $t \leq 60343 q_{H_3} q_{H_1} 2^k t/\epsilon(2^k - 1)$.*

Proof See Appendix 5. □

6 Performance discussion

We compare major computational cost, communication overhead, and formal security of straightforward "deniable authentication followed by encryption" method with those

Table 1 Performance comparison

Schemes	Computational cost			Communication overhead	Formal security
	PM	EC	PC		
SL+BF	6	3	7	$3\|G_1\| + \|\mathbb{Z}_q^*\| + \|MAC\| + 2\|m\|$	No
LXJ+BF	5	1	4	$2\|G_1\| + \|G_2\| + 2\|m\|$	No
Ours	4	1	3	$\|G_1\| + \|G_2\| + \|m\|$	Yes

of our scheme in Table 1. For convenience, we use SL+BF to denote the straightforward method based on the deniable authentication in [22] and the encryption scheme in [5] and LXJ+BF to denote the straightforward method based on the deniable authentication in [24] and the encryption scheme in [5]. We denote by PM the point multiplication in G_1, EC the exponentiation computation in G_2 and PC the pairing computation. The other operations are ignored in Table 1 since these operations take the most running time of the whole algorithm. $|x|$ denotes the number of bits of x. From Table 1, we find that our scheme has the least computational cost and communication overhead. In addition, our scheme has the formal security proof and the straightforward method has no such property.

We give a quantitative analysis for the computational cost and communication overhead. We use PBC Type A pairing [25] in this analysis. The Type A pairing is constructed on the curve

$$y^2 \equiv (x^3 + x) \bmod p$$

for some prime $p \equiv 3 \bmod 4$, where the embedding degree is 2 and the order of G_1 is q. In this analysis, we use three kinds of parameters that represents 80-bit, 112-bit and 128-bit AES [26] key size security level, respectively. Table 2 gives the specification for different security level of this analysis.

Figure 2 gives the implementation time of SL+BF, LXJ+BF and our scheme on ThinkPad T430s that is equipped with an Intel Core i5 3210M 2.5 GHz machine with 4G RAM. From Fig. 2, we know that our scheme is $\frac{169.8-83.7}{169.8} = 50.7\,\%$ faster and $\frac{108.3-83.7}{108.3} = 22.7\,\%$ faster than SL+BF and LXJ+BF, respectively, at 80-bit security level. At 112-bit security level, our scheme is $\frac{835.5-407.2}{835.5} = 51.2\,\%$ faster and $\frac{528.2-407.2}{528.2} = 22.9\,\%$ faster than SL+BF and

Table 2 Specification for different security level of this analysis (bits)

Security level	Size of p	Size of q
80-bit	512	160
112-bit	1024	224
128-bit	1536	256

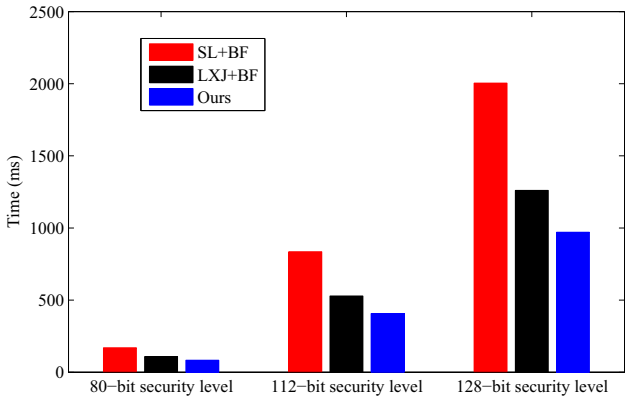


Fig. 2 The implementation time

LXJ+BF, respectively. At 128-bit security level, our scheme is $\frac{2003.9-970.3}{2003.9} = 51.5\,\%$ faster and $\frac{1260.6-970.3}{1260.6} = 23.0\,\%$ faster than SL+BF and LXJ+BF, respectively.

Now we consider the communication overhead. Note that for 80-bit, 112-bit and 128-bit security level, the corresponding output sizes of MAC are 160 bits, 224 bits and 256 bits, respectively. When we adopt the 80-bit security level, the size of p is 512 bits. So the size of an element in group G_1 is 1024 bits using an elliptic curve with 160 bits q. By standard compression technique [27], the size of an element in group G_1 can be reduced to 65 bytes. The size of an element in G_2 is 1024 bits. So, the communication overhead of SL+BF, LXJ+BF and our scheme are $3|G_1| + |\mathbb{Z}_q^*| + |MAC| + 2|m|$ bits $= 3 * 65 + 20 + 20 + |m|/4$ bytes $= 235 + |m|/4$ bytes, $2|G_1| + |G_2| + 2|m|$ bits $= 2 * 65 + 128 + |m|/4$ bytes $= 258 + |m|/4$ bytes, and $|G_1| + |G_2| + |m|$ bits $= 65 + 128 + |m|/8$ bytes $= 193 + |m|/8$ bytes, respectively. We can use the same method to compute the communication overhead at the 112-bit security level and 128-bit security level. We summarize the communication overhead at different security level in Fig. 3. Compared with SL+BF and LXJ+BF, the communication overhead (here we assume that $|m| = 1000$ bits) of our scheme is respectively reduced by $\frac{485-318}{485} = 34.4\,\%$ and $\frac{508-318}{508} = 37.4\,\%$ at the 80-bit security level. At the 112-bit security level, the communication overhead of our scheme is respectively reduced by $\frac{693-510}{693} = 26.4\,\%$ and $\frac{764-510}{764} = 33.2\,\%$. At the 128-bit security level, the communication overhead of our

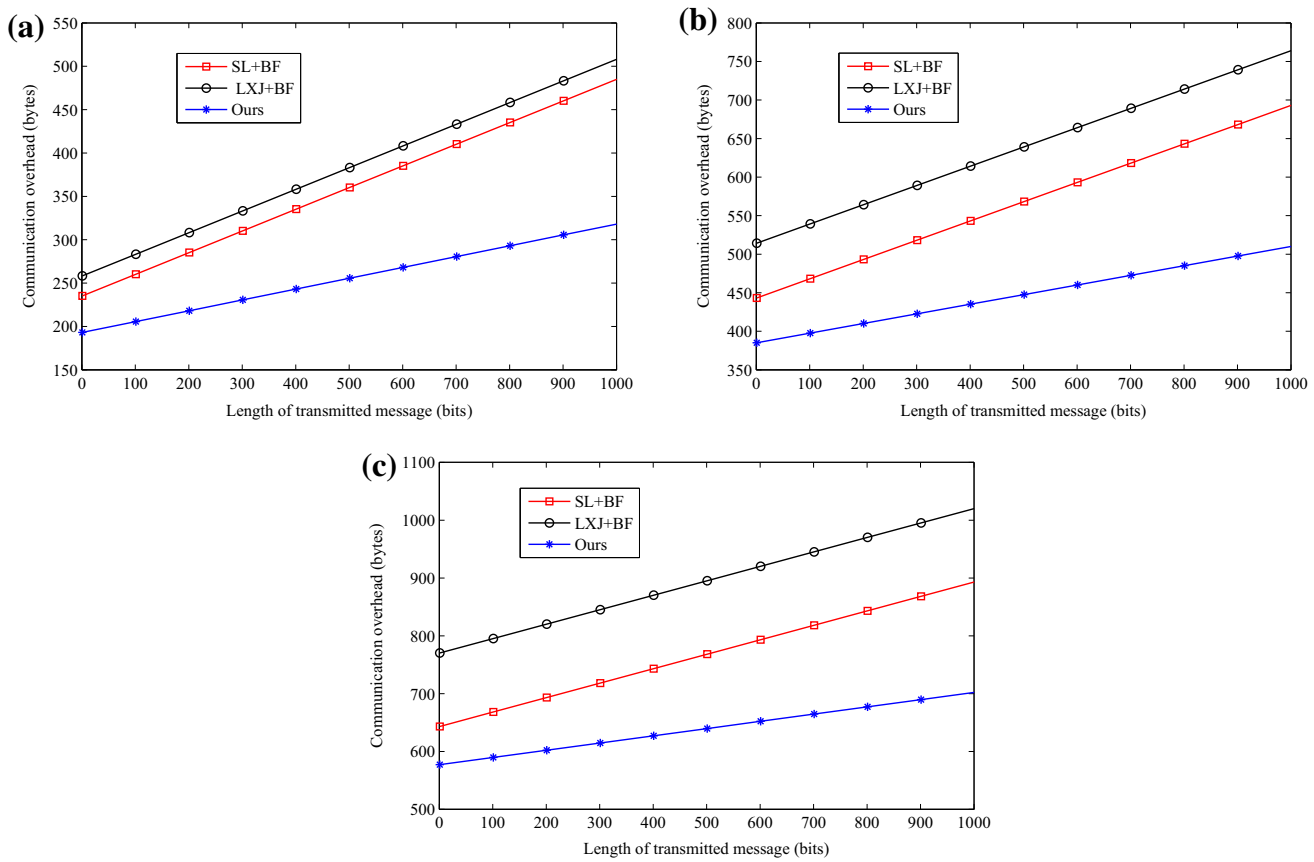


Fig. 3 The communication overhead versus length of transmitted message. **a** 80-bit security level. **b** 112-bit security level. **c** 128-bit security level

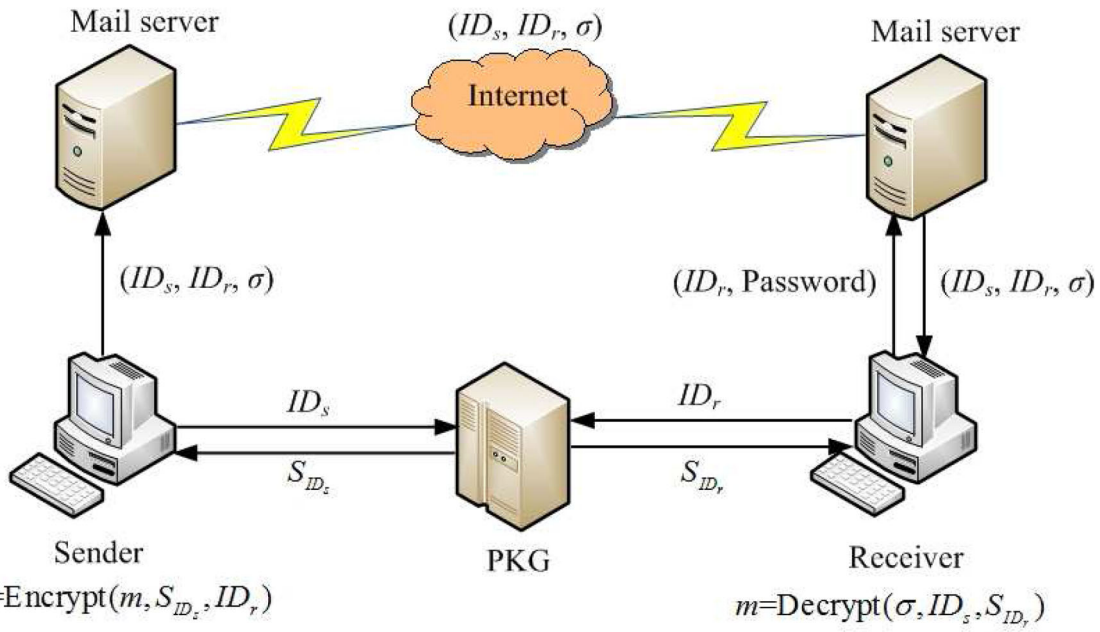


Fig. 4 A secure e-mail protocol

scheme is respectively reduced by $\frac{893-702}{893} = 21.3\,\%$ and $\frac{1020-702}{1020} = 31.1\,\%$.

7 Application

In this section, we show an application of IBDEA to an e-mail system. A secure e-mail system should provides confidentiality and authentication. If we adopt "digital signature followed by encryption" method, the privacy of the sender may be infringed since the receiver can prove the source to any third party. The DAE solves this problem in a good way. Here we give a secure e-mail protocol using our IBDAE. This protocol consists of a sender, a receiver and mail servers and is described in Fig. 4. In this protocol, the identities of the sender and receiver are e-mail addresses. For example, the sender's identity ID_s is `alice@uestc.edu.cn` and the receiver's identity ID_r is `bob@uestc.edu.cn`. The sender first applies a private key S_{ID_s} from the PKG and then runs $\texttt{Encrypt}(m, S_{ID_s}, ID_r)$ to get a ciphertext σ. The sender transmits its identity ID_s, the receiver's identity ID_r and the ciphertext σ to its mail server. Then the sender's mail server transfers (ID_s, ID_r, σ) to the receiver's mail server. The receiver's mail server keeps (ID_s, ID_r, σ) and waits for the receiver. When the receiver hopes to receive its mail, it submits its identity ID_r and password to its mail server for authentication. If the receiver passes this authentication, the mail server sends the (ID_s, ID_r, σ) to the receiver. The receiver can apply a private key S_{ID_r} from the PKG and then runs $\texttt{Decrypt}(\sigma, ID_s, S_{ID_r})$ to obtain the message m. Of course, the receiver can registers a private key S_{ID_s} in advance. The hybrid technique is very suitable for sending a large e-mail.

8 Conclusions

In this paper, we proposed a hybrid IBDAE scheme that can achieve confidentiality and deniable authentication in a logical single step. Our construction is based on an IBDATK scheme and a DEM scheme. We proposed an IBDATK scheme using bilinear pairings and proved its security in the random oracle model. For typical security level, our scheme is at least 50.7 and 22.7 % faster than two straightforward "deniable authentication followed by encryption" schemes, respectively. The communication overhead is respectively reduced at least 21.3 and 31.1 %. A secure e-mail protocol based on IBDAE was designed.

Acknowledgments This work is supported by the National Natural Science Foundation of China (Grant Nos. 61073176, 61272525, 61302161 and 61462048) and the Fundamental Research Funds for the Central Universities (Grant No. ZYGX2013J069).

Appendix

Data encapsulation mechanism (DEM)

A DEM is a symmetric encryption scheme that consists of the following two algorithms.

- `Enc` is a deterministic encryption algorithm that takes as input a security parameter k, a key K and a message $m \in \{0,1\}^*$, and outputs a ciphertext $c \in \{0,1\}^*$, where $K \in \mathcal{K}_{\text{DEM}}$ is a key in the given key space, and m is a bit string of arbitrary length. We denote this as $c = \texttt{Enc}(K, m)$.
- `Dec` is a deterministic decryption algorithm that takes as input a key K and a ciphertext c, and outputs the message $m \in \{0,1\}^*$ or a symbol $\perp$ to indicate that the ciphertext is invalid.

In this paper, we only require that a DEM is secure with respect to indistinguishability against passive attackers (IND-PA). Formally, this security notion is captured by the following game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- `Initial` $\mathcal{A}$ runs on input k and submits two equal length messages, m_0 and m_1.
- `Challenge` $\mathcal{C}$ selects a random key $K \in \mathcal{K}_{\text{DEM}}$ and a random bit $\beta \in \{0,1\}$, and sends $c = \texttt{Enc}(K, m_\beta)$ to $\mathcal{A}$ as a challenge ciphertext.
- `Guess` $\mathcal{A}$ outputs a bit β' and wins the game if $\beta' = \beta$.

The advantage of $\mathcal{A}$ is defined to be $\text{Adv}(\mathcal{A}) = |2\Pr[\beta' = \beta] - 1|$, where $\Pr[\beta' = \beta]$ is the probability that $\beta' = \beta$.

Definition 7 An adversary $\mathcal{A}$ is said to be an (ϵ_{dem}, t)-attacker of a DEM scheme if $\mathcal{A}$ has advantage at least ϵ_{dem} in the above game, runs in time at most t. A DEM scheme is said to be (ϵ_{dem}, t)-IND-PA secure if no (ϵ_{dem}, t)-attacker exists.

Proof of Theorem 1

Proof Our proof method is as follows. We define a sequence $\texttt{Game}_0, \texttt{Game}_1, \texttt{Game}_2$ of modified attack games. The only difference between games is how the environment responds to $\mathcal{A}$'s oracle queries.

Let $\sigma^* = (\psi^*, c^*)$ be the challenge ciphertext submitted to $\mathcal{A}$ by its challenge oracle that encrypts either m_0 or m_1 according to a bit β. Let K^* denote the symmetric key used by the challenge oracle in the generation of the challenge ciphertext, or alternatively, the decapsulation of ψ^* using the identities ID_s and ID_r that are chosen by the adversary. For any $i = 0, 1, 2$, we let S_i be the event that $\delta' = \delta$ in game $\mathtt{Game}_i$, where δ is the bit chosen by $\mathcal{A}$'s challenge oracle and δ' is the bit output by $\mathcal{A}$. This probability is taken over the random choices of $\mathcal{A}$ and those of $\mathcal{A}$'s oracles.

We will use the following useful lemma from [28].

Lemma 1 ([28]) *Let E, E', and F be events defined on a probability space such that* $\Pr[E \wedge \neg F] = \Pr[E' \wedge \neg F]$. *Then we have*

$$|\Pr[E] - \Pr[E']| \leq \Pr[F].$$

$\mathtt{Game}_0$: We simulate the view of the adversary in a real attack by running the key extraction algorithm and using the resulting keys to respond to $\mathcal{A}$'s queries. So the view of $\mathcal{A}$ is the same as it would be in a real attack. Therefore, we have

$$|\Pr[S_0] - \frac{1}{2}| = \frac{1}{2}\epsilon_{dae}.$$

$\mathtt{Game}_1$: In this game, we slightly modify how the decryption oracle responds to queries from $\mathcal{A}$. When a ciphertext $\sigma = (\psi, c)$, a sender' identity ID_i and a receiver's identity ID_j are presented to the decryption oracle after the invocation of the challenge encryption oracle, if $ID_i = ID_s$, $ID_j = ID_r$ and $\psi = \psi^*$, then the decryption oracle does not use the genuine decryption procedure for the hybrid scheme, instead it uses the key K^* to decrypt c and returns the result to the adversary $\mathcal{A}$.

Obviously this change has no impact on the adversary and so

$$\Pr[S_1] = \Pr[S_0].$$

$\mathtt{Game}_2$: In this game, we modify Game_1 by replacing K^* with a random key K' from $\mathcal{K}_{\mathrm{DEM}}$. The result then follows from the following Lemmas 2 and 3. □

Lemma 2 *There exists a probabilistic polynomial-time algorithm $\mathcal{C}_1$, whose running time is essentially the same as that of $\mathcal{A}$, such that*

$$|\Pr[S_2] - \Pr[S_1]| = \epsilon_{datk}.$$

Proof In this proof, we show how to construct an adversary $\mathcal{C}_1$ of the IBDATK to violate the assumed security against adaptive chosen ciphertext attack.

- `Initial` Given the system parameters *param*, $\mathcal{C}_1$ sends it to $\mathcal{A}$.
- `Phase 1` When $\mathcal{A}$ makes a key extraction query on identity ID, $\mathcal{C}_1$ makes a key extraction query to its own key extraction oracle and forwards the answer to $\mathcal{A}$. When $\mathcal{A}$ makes an encryption query with a plaintext m, a sender's identity ID_i and a receiver's identity ID_j, $\mathcal{C}_1$ follows the steps below.
  1. Make a symmetric key generation query on (ID_i, ID_j) to its own symmetric key generation oracle to obtain K.
  2. Compute $c = \mathtt{DEM.Enc}(K, m)$.
  3. Make a key encapsulation query on c to its own key encapsulation oracle to obtain ψ.
  4. Return the ciphertext $\sigma = (\psi, c)$ to $\mathcal{A}$.

  When $\mathcal{A}$ makes a decryption query with a ciphertext $\sigma = (\psi, c)$, a sender's identity ID_i and a receiver's identity ID_j, $\mathcal{C}_1$ follows the steps below.
  1. Make a key decapsulation query on (ψ, c, ID_i, ID_j) to its own key decapsulation oracle to obtain K.
  2. If $K = \perp$, return $\perp$ and stop.
  3. Compute $m = \mathtt{DEM.Dec}(K, c)$ and return m.
- `Challenge` $\mathcal{A}$ generates two equal length plaintexts m_0 and m_1, a sender's identity ID_s and a receiver's identity ID_r on which it wishes to be challenged. $\mathcal{C}_1$ follows the steps below.
  1. Submit ID_s and ID_r to its challenger to obtain K_β, where $\beta \in \{0, 1\}$.
  2. Pick a random bit δ from $\{0, 1\}$.
  3. Compute $c^* = \mathtt{DEM.Enc}(K_\beta, m_\delta)$.
  4. Submit c^* to its challenger to obtain ψ^*.
  5. Return the ciphertext $\sigma^* = (\psi^*, c^*)$ to $\mathcal{A}$.
- `Phase 2` $\mathcal{A}$ can ask a polynomially bounded number of queries adaptively again as in phase 1 with the restriction that it cannot make a key extraction query on ID_r and it cannot make a decryption query on $\sigma^* = (\psi^*, c^*)$ to obtain the corresponding plaintext.
- `Guess` $\mathcal{A}$ outputs δ'. If $\delta' = \delta$, $\mathcal{C}_1$ outputs $\beta' = 1$ indicating K_β is the real key; otherwise it outputs $\beta' = 0$ indicating K_β is a random key.

When K_β is the real key, $\mathcal{A}$ is run exactly as it would be run in $\mathtt{Game}_1$. This means that

$$\Pr[S_1] = \Pr[\delta' = \delta | \beta = 1] = \Pr[\beta' = 1 | \beta = 1].$$

When K_β is the random key, $\mathcal{A}$ is run exactly as it would be in $\mathtt{Game}_2$. This means that

$$\Pr[S_2] = \Pr[\delta' = \delta | \beta = 0] = \Pr[\beta' = 1 | \beta = 0].$$

From the definition of security for IBDATK, we have

$$\epsilon_{datk} = |2\Pr[\beta' = \beta] - 1| = |\Pr[\beta' = 1|\beta = 1] - \Pr[\beta' = 1|\beta = 0]|.$$

So the result holds. □

Lemma 3 *There exists a probabilistic polynomial-time algorithm $\mathcal{C}_2$, whose running time is essentially the same as that of $\mathcal{A}$, such that*

$$|\Pr[S_2] - \frac{1}{2}| = \frac{1}{2}\epsilon_{dem}.$$

Proof To construct such a $\mathcal{C}_2$ we simply run $\mathcal{A}$ as it would be run in game $\mathtt{Game}_2$. We run the key extraction step so we can respond to $\mathcal{A}$'s queries before it calls its challenge encryption oracle. When $\mathcal{A}$ calls its challenge encryption oracle with two messages m_0 and m_1, a sender's identity ID_s and a receiver's identity ID_r, we simply relay m_0 and m_1 to the challenge encryption oracle of $\mathcal{C}_2$ to obtain c^*. We then make a symmetric key generation query and a key encapsulation query to obtain K^* and ψ^*, respectively. We discard K^* and return $\sigma^* = (\psi^*, c^*)$ to $\mathcal{A}$.

In this simulation $\mathcal{A}$ is run by $\mathcal{C}_2$ in exactly the same manner as the former would be run in game $\mathtt{Game}_2$; moreover, $\Pr[S_2]$ corresponds exactly to the probability that $\mathcal{C}_2$ correctly determines the hidden bit of its challenge encryption oracle since $\mathcal{C}_2$ outputs whatever $\mathcal{A}$ outputs. The result follows. □

Proof of Theorem 2

Proof Suppose that $\mathcal{F}$ is an adversary who can breaks the DA-CMA security of the IBDAE scheme with advantage ϵ_{dae}. We can use this to construct another algorithm $\mathcal{C}$ that can break the DA-CMA security of the IBDATK with advantage at least ϵ_{datk} too. $\mathcal{C}$ runs as follow.

- Initial Given the system parameters *param*, $\mathcal{C}$ sends it to $\mathcal{F}$.
- Attack When $\mathcal{F}$ makes a key extraction query on identity ID, $\mathcal{C}$ makes a key extraction query to its own key extraction oracle and forwards the answer to $\mathcal{F}$. When $\mathcal{F}$ makes an encryption query with a plaintext m, a sender's identity ID_i and a receiver's identity ID_j, $\mathcal{C}$ follows the steps below.
  1. Make a symmetric key generation query on (ID_i, ID_j) to its own symmetric key generation oracle to obtain K.
  2. Compute $c = \mathtt{DEM.Enc}(K, m)$.
  3. Make a key encapsulation query on c to its own key encapsulation oracle to obtain ψ.
  4. Return the ciphertext $\sigma = (\psi, c)$ to $\mathcal{F}$.

  When $\mathcal{F}$ makes a decryption query with a ciphertext $\sigma = (\psi, c)$, a sender's identity ID_i and a receiver's identity ID_j, $\mathcal{C}$ follows the steps below.
  1. Make a key decapsulation query on (ψ, c, ID_i, ID_j) to its own key decapsulation oracle to obtain K.
  2. If $K = \perp$, return $\perp$ and stop.
  3. Compute $m = \mathtt{DEM.Dec}(K, c)$ and return m.
- Forgery $\mathcal{F}$ outputs (σ^*, ID_s, ID_r), where $\sigma^* = (\psi^*, c^*)$. $\mathcal{C}$ outputs $(\psi^*, \tau^*, ID_s, ID_r)$, where $\tau^* = c^*$.

Obviously, this simulation is perfect. If $\mathcal{F}$ wins the DA-CMA game for the IBDAE, $\mathcal{C}$ have the same probability to win the DA-CMA game for the IBDATK. □

Proof of Theorem 3

Proof $\mathcal{C}$ receives a random instance (P, aP, bP, cP, θ) of the DBDH problem and attempts to decide whether $\theta = \hat{e}(P, P)^{abc}$ or not. $\mathcal{C}$ will run $\mathcal{A}$ as a subroutine and play $\mathcal{A}$'s challenger in Definition 3. $\mathcal{A}$ will consult $\mathcal{C}$ for answers to the random oracles H_1, H_2 and H_3. Roughly speaking, these answers are randomly generated, but to maintain the consistency and to avoid collision. $\mathcal{C}$ keeps three lists L_1, L_2 and L_3 respectively to store the answers. The following assumptions are made.

1. $\mathcal{A}$ will ask for $H_1(ID)$ before ID is used in any key extraction queries, symmetric key generation queries, key encapsulation queries and key decapsulation queries.
2. An encapsulation returned from a key encapsulation query will not be used by $\mathcal{A}$ in a key decapsulation query.
3. The sender's identity and the receiver's identity are different in Encap and Decap algorithms. This assumption is called irreflexivity assumption in [11]. Such an assumption makes our proof simple.

- Initial $\mathcal{C}$ gives $\mathcal{A}$ the system parameters with $P_{pub} = cP$. Note that c is unknown to $\mathcal{C}$. This value simulates the master secret key for the PKG.
- Phase 1 $\mathcal{A}$ performs a polynomially bounded number of queries.
  - *H_1 queries* $\mathcal{C}$ first selects a number $\lambda \in \{1, 2, \ldots, q_{H_1}\}$ randomly. $\mathcal{A}$ asks a polynomially bounded number of H_1 queries on identities of its choice. At the λ-th H_1 query, $\mathcal{C}$ answers by $H_1(ID_\lambda) = bP$. For queries $H_1(ID_i)$ with $i \neq \lambda$, $\mathcal{C}$ selects $d_i \in \mathbb{Z}_q^*$ randomly, inserts the pair (ID_i, d_i) in the list L_1 and answers $H_1(ID_i) = d_i P$.
  - *H_2 and H_3 queries* When $\mathcal{A}$ asks queries on these hash values, $\mathcal{C}$ checks the corresponding list. If an entry for the query is found, the same answer will be

given to $\mathcal{A}$; otherwise, a randomly chosen value will be used as the answer to $\mathcal{A}$. The query and answer will then be inserted in the list.
- *Key extraction queries* $\mathcal{A}$ submits an identity ID_i to $\mathcal{C}$. If $ID_i = ID_\lambda$, then $\mathcal{C}$ fails and stops. If $ID_i \neq ID_\lambda$, then the list L_1 must contain a pair (ID_i, d_i) for some d_i (this indicates $\mathcal{C}$ previously answered $H_1(ID_i) = d_i P$). The private key for ID_i is $d_i P_{pub} = d_i c P$. It is computed by $\mathcal{C}$ and returned to $\mathcal{A}$.
- *Symmetric key generation queries* $\mathcal{A}$ produces a sender's identity ID_i and a receiver's identity ID_j. If $ID_i \neq ID_\lambda$. $\mathcal{C}$ first obtains the private key S_{ID_i} by running a key extraction query. Then $\mathcal{C}$ answers the query by running $(K, \omega) = \mathtt{Sym}(S_{ID_i}, ID_j)$ algorithm. Note that $\mathcal{C}$ needs to store ω and to overwrite any previous value. If $ID_i = ID_\lambda$ and hence $ID_j \neq ID_\lambda$ by the irreflexivity assumption, $\mathcal{C}$ first chooses x from $\mathbb{Z}_q^*$ randomly and computes $z = \hat{e}(P_{pub}, Q_{ID_j})^x$. Then $\mathcal{C}$ runs the H_2 simulation algorithm to find $K = H_2(z)$. Finally, $\mathcal{C}$ gives K to $\mathcal{A}$. Note that the ω is (x, z, ID_i, ID_j) in this case.
- *Key encapsulation queries* $\mathcal{A}$ produces an arbitrary tag τ. $\mathcal{C}$ checks whether there exists a stored value ω. If not, it returns $\perp$ and stops. Otherwise we have the following two cases to consider. The first case is $ID_i \neq ID_\lambda$. In this case, $\mathcal{C}$ just runs $\psi = \mathtt{Encap}(\omega, \tau)$ and returns the result to $\mathcal{A}$. The second case is $ID_i = ID_\lambda$ and hence $ID_j \neq ID_\lambda$ by the irreflexivity assumption. In this case, $\mathcal{C}$ first runs the H_3 simulation algorithm to get $u = H_3(\tau, z)$ and then computes $R = uQ_{ID_i}$ and $T = z \cdot \hat{e}(R, S_{ID_j})$($\mathcal{C}$ could obtain S_{ID_j} from the key extraction algorithm because $ID_j \neq ID_\lambda$). Finally, $\mathcal{C}$ gives $\psi = (R, T)$ to $\mathcal{A}$.
- *Key decapsulation queries* $\mathcal{A}$ selects an encapsulation ψ, a tag τ, a sender's identity ID_i and a receiver's identity ID_j. If $ID_j = ID_\lambda$, $\mathcal{C}$ always answers $\mathcal{A}$ that (ψ, τ) is invalid. If $\mathcal{A}$ previously asked the hash value $H_3(\tau, z)$, where $z = T/\hat{e}(R, S_{ID_j})$, and $\mathcal{C}$ answered u that satisfies $R = uQ_{ID_i}$, $\mathcal{C}$ will fail. In this instance, (ψ, τ) is actually valid from $\mathcal{A}$'s point of view. The probability of this instance is at most $1/2^k$. If $ID_j \neq ID_\lambda$, $\mathcal{C}$ first obtains S_{ID_j} from the key extraction algorithm and then sends the result of $\mathtt{Decap}(\psi, \tau, ID_i, S_{ID_j})$ to $\mathcal{A}$.

– `Challenge` $\mathcal{A}$ decides when phase 1 ends. $\mathcal{A}$ produces a sender's identity ID_s and a receiver's identity ID_r on which it hopes to be challenged. If $\mathcal{A}$ has asked a key extraction query on ID_λ, $\mathcal{C}$ fails. On the other hand, if $\mathcal{A}$ does not choose $ID_r = ID_\lambda$ as target identity, $\mathcal{C}$ fails too. $\mathcal{C}$ first chooses $T^* \in G_2$, sets $R^* = aP$ and computes $z^* = T^*/\theta$ (where θ is $\mathcal{C}$'s candidate for the DBDH problem). Then $\mathcal{C}$ runs the H_2 simulation algorithm to find $K_1 = H_2(z^*)$. $\mathcal{C}$ chooses $K_0 \in \mathcal{K}_{\mathrm{IBDATK}}$ and a bit $\beta \in \{0, 1\}$ randomly, and sends K_β to $\mathcal{A}$. $\mathcal{A}$ then sends a tag τ^* to $\mathcal{C}$. Finally, $\mathcal{C}$ gives the encapsulation $\psi^* = (R^*, T^*)$ to $\mathcal{A}$.
– `Phase 2` $\mathcal{A}$ can ask a polynomially bounded number of queries adaptively again as in phase 1 with the restriction that it cannot make a key extraction query on ID_r and cannot make a decapsulation query on (ψ^*, τ^*) to obtain the corresponding key.
– `Guess` $\mathcal{A}$ outputs a bit β' for which it believes the relation $(K_\beta, \omega^*) = \mathtt{Sym}(S_{ID_s}, ID_r)$ and $\psi^* = \mathtt{Encap}(\omega^*, \tau^*)$ hold. At this moment, if $\beta' = \beta$, $\mathcal{C}$ returns 1 that denotes $\theta = \hat{e}(P, P)^{abc}$; otherwise, it returns 0 that denotes $\theta \neq \hat{e}(P, P)^{abc}$.

We now assess $\mathcal{C}$'s probability of success. Note that $\mathcal{C}$ only fails in providing a consistent simulation because one of the following independent events:

E_1 $\mathcal{A}$ does not choose ID_λ as the receiver's identity in the challenge phase.
E_2 $\mathcal{A}$ has made a key extraction query on ID_λ.
E_3 $\mathcal{C}$ aborts in a key decapsulation query because of rejecting an valid encapsulation.

We know that $\Pr[\neg E_1] = 1/q_{H_1}$ and $\Pr[E_3] \leq q_d/2^k$. In addition, we know that $\neg E_1$ implies $\neg E_2$.

Since

$$\begin{aligned} p_1 &= \Pr[\beta' = \beta | (K_\beta, \omega^*) = \mathtt{Sym}(S_{ID_s}, ID_r) \text{ and } \psi^* \\ &\quad = \mathtt{Encap}(\omega^*, \tau^*)] \\ &= \frac{\epsilon + 1}{2} - \frac{q_d}{2^k} \end{aligned}$$

and

$$p_0 = \Pr[\beta' = i | \theta \in_R G_2] = \frac{1}{2} \quad \text{for} \quad i = 0, 1,$$

We have

$$\begin{aligned} \mathrm{Adv}(\mathcal{C}) &= \frac{|p_1 - p_0|}{q_{H_1}} = \left(\frac{\epsilon + 1}{2} - \frac{q_d}{2^k} - \frac{1}{2}\right)\left(\frac{1}{q_{H_1}}\right) \\ &= \frac{\epsilon - q_d/2^{k-1}}{2q_{H_1}} \end{aligned}$$

The bound on $\mathcal{C}$'s computation time comes from the fact that $\mathcal{C}$ needs $O(q_s + q_e + q_d)$ pairing operations in the symmetric key generation queries, key encapsulation queries and key decapsulation queries. □

Proof of Theorem 4

Proof Here we use the forking lemma [29] to prove the security of our scheme. To use the forking lemma, we need to show how our scheme fits into the signature scheme described in [29], the simulation step in which the signature can be simulated without the sender's private key (and thus, also without the master secret key), and how we can solve BDH problem based on the forgery.

First, we observe that our scheme satisfies the requirement described in [29]. During the encapsulation of a tag τ, the tuple (σ_1, u, σ_2) is generated that corresponds to the required three-phase honest-verifier zero-knowledge identification protocol, where $\sigma_1 = z$ is the commitment of the prover, $u = H_3(\tau, z)$ is the hash value depending on τ and σ_1 substituted for the verifier's challenge, and $\sigma_2 = T$ is the response of the prover that relies on σ_1, u and the sender's private key S_{ID_s}.

Next, we need to show a simulation step that supplies a faithful simulation to the forger $\mathcal{F}$ and how to solve the BDH problem by interacting with $\mathcal{F}$. $\mathcal{C}$ receives a random instance (P, aP, bP, cP) of the BDH problem and is required to compute $h = \hat{e}(P, P)^{abc}$. $\mathcal{C}$ will run $\mathcal{F}$ as a subroutine and plays $\mathcal{F}$'s challenger in the game of Definition 4. We describe this process as follows.

- `Initial` $\mathcal{C}$ runs `Setup` algorithm with a security parameter k and sends the system parameters with $P_{pub} = cP$ to $\mathcal{F}$.
- `Attack` $\mathcal{C}$ answers $\mathcal{F}$'s queries according to the method in Theorem 3 except that the simulator should choose two different random numbers $\chi, \xi \in \{1, 2, \ldots, q_{H_1}\}$ in advance. At the χ-th H_1 query, $\mathcal{C}$ answers by $H_1(ID_\chi) = aP$. At the ξ-th H_1 query, $\mathcal{C}$ answers by $H_1(ID_\xi) = bP$. If we let $ID_\lambda = \{ID_\chi, ID_\xi\}$, the simulation process is the same as that in Theorem 3.
- `Forgery` $\mathcal{F}$ outputs a quaternion $(\psi^*, \tau^*, ID_\chi, ID_\xi)$, where $\psi^* = (R^*, T^*)$. We coalesce the identities $ID_\lambda = \{ID_\chi, ID_\xi\}$ and the tag τ^* into a "generalized" forged tag (ID_λ, τ^*) so as to hide the identity-based aspect of the DA-CMA attack, and simulate the setting of an identityless adaptive-CMA existential forgery for which the forking lemma is proven. From the the forking lemma [29], if $\mathcal{F}$ is a sufficiently efficient forger in the above interaction, then we can construct a Las Vegas machine $\mathcal{F}'$ that outputs two signed tags $((ID_\lambda, \tau^*), u^*, \psi^*)$ and $((ID_\lambda, \tau^*), \bar{u}^*, \bar{\psi}^*)$ with $u^* \neq \bar{u}^*$ and the same commitment z^*. To solve the BDH problem based on the machine $\mathcal{F}'$ derived from $\mathcal{F}$, we construct a machine $\mathcal{C}'$ as follows.

  1. $\mathcal{C}'$ runs $\mathcal{F}'$ to get two distinct signatures $((ID_\lambda, \tau^*), u^*, \psi^*)$ and $((ID_\lambda, \tau^*), \bar{u}^*, \bar{\psi}^*)$.
  2. $\mathcal{C}'$ computes $\hat{e}(P, P)^{abc}$ as $(T^*/\bar{T}^*)^{1/(u^*-\bar{u}^*)}$.

From the forking lemma [29] and the lemma on the relationship between given-identity attack and chosen-identity attack [8], if $\mathcal{F}$ succeeds in time t with probability $\epsilon \geq 5(q_e+1)(q_e+q_{H_3})q_{H_1}/(2^k-1)$, then $\mathcal{C}'$ can solve the BDH problem in expected time $t \leq 60343 q_{H_3} q_{H_1} 2^k t/\epsilon(2^k-1)$. Note that the coefficient is changed because the simulator should choose two different identities in advance. □

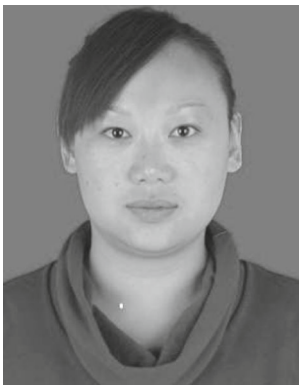
References

1. Maimut, D., & Reyhanitabar, R. (2014). Authenticated encryption: Toward next-generation algorithms. *IEEE Security & Privacy*, *12*(2), 70–72.
2. Wu, T. S., & Lin, H. Y. (2014). Provably secure proxy convertible authenticated encryption scheme based on RSA. *Information Sciences*, *278*, 577–587.
3. Xie, Q. (2012). Provably secure convertible multi-authenticated encryption scheme. *IET Information Security*, *6*(2), 65–70.
4. Shamir, A. (1984). Identity-based cryptosystems and signature schemes. In *Proc CRYPTO'84, LNCS* (Vol. 196, pp. 47–53). Springer.
5. Boneh, D., & Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Proc CRYPTO 2001, LNCS* (Vol. 2139, pp. 213–229). Springer.
6. Park, J. H., & Lee, D. H. (2015). An efficient IBE scheme with tight security reduction in the random oracle model. *Designs, Codes and Cryptography*. doi:10.1007/s10623-015-0035-0.
7. Hess, F. (2003). Efficient identity based signature schemes based on pairings. In *Proc SAC 2002, LNCS* (Vol. 2595, pp. 310–324). Springer.
8. Cha, J. C., & Cheon, J. H. (2003). An identity-based signature from gap Diffie–Hellman groups. In *Proc PKC 2003, LNCS* (Vol. 2567, pp. 18–30). Springer.
9. Hsu, C. L., & Lin, H. Y. (2011). New identity-based key-insulated convertible multi-authenticated encryption scheme. *Journal of Network and Computer Applications*, *34*(5), 1724–1731.
10. Lin, H. Y., & Hsu, C. L. (2011). A novel identity-based key-insulated convertible authenticated encryption scheme. *International Journal of Foundations of Computer Science*, *22*(3), 739–756.
11. Boyen, X. (2003). Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography. In *Proc CRYPTO 2003, LNCS* (Vol. 2729, pp. 383–399). Springer.
12. Barreto, P. S. L. M., Libert, B., McCullagh, N., & Quisquater, J. J. (2005). Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Proc ASIACRYPT 2005, LNCS* (Vol. 3788, pp. 515–532). Springer.
13. Lin, C., Tang, F., Ke, P., Harn, L., & Zhang, S. (2014). Secure universal designated verifier identity-based signcryption. *Security and Communication Networks*, *7*(2), 434–444.
14. Cramer, R., & Shoup, V. (2003). Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, *33*(1), 167–226.
15. Abe, M., Gennaro, R., & Kurosawa, K. (2008). Tag-KEM/DEM: A new framework for hybrid encryption. *Journal of Cryptology*, *21*(1), 97–130.
16. Manulis, M., Poettering, B., & Stebila, D. (2014). Plaintext awareness in identity-based key encapsulation. *International Journal of Information Security*, *13*(1), 25–49.

17. Kang, L., Tang, X. H., & Liu, J. F. (2014). Tight chosen ciphertext attack (CCA)-secure hybrid encryption scheme with full public verifiability. *SCIENCE CHINA Information Sciences*, *57*(11), 1–14.
18. Bentahar, K., Farshim, P., Malone-Lee, J., & Smart, N. P. (2008). Generic constructions of identity-based and certificateless KEMs. *Journal of Cryptology*, *21*(2), 178–199.
19. Abdalla, M., Catalano, D., & Fiore, D. (2014). Verifiable random functions: Relations to identity-based key encapsulation and new constructions. *Journal of Cryptology*, *27*(3), 544–593.
20. Raimondo, M. D., & Gennaro, R. (2009). New approaches for deniable authentication. *Journal of Cryptology*, *22*(4), 572–615.
21. Li, F., & Takagi, T. (2013). Cryptanalysis and improvement of robust deniable authentication protocol. *Wireless Personal Communications*, *69*(4), 1391–1398.
22. Shi, Y., & Li, J. (2005). Identity-based deniable authentication protocol. *Electronics Letters*, *41*(5), 241–242.
23. Kar, J. (2013). ID-based deniable authentication protocol based on Diffie–Hellman problem on elliptic curve. *International Journal of Network Security*, *15*(5), 357–364.
24. Li, F., Xiong, P., & Jin, C. (2014). Identity-based deniable authentication for ad hoc networks. *Computing*, *96*(9), 843–853.
25. PBC Library. http://crypto.stanford.edu/pbc/.
26. Daemen, J., & Rijmen, V. (2002). *The design of Rijndael: AES—The Advanced Encryption Standard*. Berlin: Springer.
27. Shim, K. A. (2012). CPAS: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. *IEEE Transactions on Vehicular Technology*, *61*(4), 1874–1883.
28. Shoup, V. (2001). OAEP reconsidered. In *Proc CRYPTO 2001, LNCS* (Vol. 2139, pp. 239–259). Springer.
29. Pointcheval, D., & Stern, J. (2000). Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, *13*(3), 361–396.

Fagen Li is now an associate professor in the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, P.R. China. He received his Ph.D. degree in Cryptography from Xidian University, Xi'an, P.R. China in 2007. From 2008 to 2009, he was a postdoctoral fellow in Future University-Hakodate, Hokkaido, Japan, which is supported by the Japan Society for the Promotion of Science. He worked as a research fellow in the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan from 2010 to 2012. His recent research interests include cryptography and network security. He has published more than 70 papers in the international journals and conferences. He is a member of the IEEE.

Zhaohui Zheng received her B.S. degree in Computer Science from Henan Normal University, Zhengzhou, P.R. China in 2012. She is now a master student in the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, P.R. China. Her research interests include cryptography and information security.



Chunhua Jin is now a Ph.D. student in the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, P.R. China. Her research interests include cryptography and network security.