Efficient Certificateless Public Key Encryption

Zhaohui Cheng and Richard Comley

School of Computing Science, Middlesex University
White Hart Lane, London N17 8HR, United Kingdom
{m.z.cheng, r.comley}@mdx.ac.uk

Abstract. In [3] Al-Riyami and Paterson introduced the notion of "Certificateless Public Key Cryptography" and presented an instantiation. In the paper, we construct a more efficient scheme of certificateless public key encryption and extend it to an authenticated encryption.

1 Introduction

To address the threat of the man-in-the-middle attack, a public key infrastructure (PKI) managing certificates is needed to establish a secure system in the traditional public key cryptography setting. However, a PKI faces many challenges in the practice, especially the scalability of the infrastructure. In [20], Shamir first introduced the notion of identity-based cryptography (IBC) in which the identity of an entity is the public key, so to reduce the complexity of managing certificates. However, the key escrow function is integrated in this setting. To combine the advantage of both systems, in [3] Al-Riyami and Paterson brought forth the notion of "Certificateless Public Key Cryptography" (CL-PKC). So far, there are two major constructions of certificateless public key encryption (CL-PKE), i.e., the scheme proposed by Gentry in [14] and the system in [3] (we call it AP's scheme). In this paper, we present another CL-PKE scheme which is just the combination of an IBE and a traditional PKE. The new scheme is more efficient on computation or published information than the existing schemes. Meanwhile, we extend the scheme to an authenticated encryption.

The paper is organized as follows. First we rethink the formulation of CL-PKE and review a primitive and two existing CL-PKE schemes. The new CL-PKE scheme is presented in Section 3 and then we extend it to an authenticated encryption. Finally, we remark the three CL-PKE schemes on complexity and application.

2 Preliminaries

2.1 Certificateless Public Key Encryption

The basic rational behind the certificateless public key encryption is simple, i.e., even the adversary successfully replaces the victim's public key with his own choice (hence the adversary could know the corresponding private key), it still

cannot decrypt the message encrypted with the public key that it published. This will dramatically reduce the adversary's interest to launch such kind of attack, which is one of the major threats in the traditional public key systems. Although the idea is attractive, it is uninstantiable in the traditional public key system in which an entity's private key corresponds merely to the entity's public key. To get around this problem, a different methodology is adopted, i.e., to prevent the adversary from freely publishing the public key for any other entity. A trusted third party (TTP) introduced plays an active role to issue a certificate to bind a key pair with an entity. In a certificate, the identity component is merely used to identify the entity who owns the public/private key pair. While, after the first provable-practical identity-based encryption scheme finally was materialized [6], the identity of an entity could also serve as a public key and the CL-PKE becomes realistic. Here, we follow the formulation in [3] to define the CL-PKE with some simplification. A CL-PKE scheme involving a TTP (the "Private Key Generator" (PKG)) consists of following algorithms.

- Setup. This algorithm takes a security parameter k and returns **params** (system parameters) and a **master-key**. The system parameters include a description of a finite message space $\mathcal{M}$, and a description of a finite ciphertext space $\mathcal{C}$. The system parameters will be publicly known, while the **master-key** will be known only to the PKG.
- Extract. This algorithm running on the PKG takes as input **params**, the **master-key**, and a string $ID_A \in \{0,1\}^*$ from entity A, and returns a private key d_A denoted by $PrivKeyL$.
- Publish. This algorithm taking as input **params**, returns a private key t_A denoted by $PrivKeyR$ and the public key N_A for an entity A.
- Encrypt. This algorithm takes as input **params**, the identity ID_A of entity A, a message $m \in \mathcal{M}$ and the public key N_A of A and returns a ciphertext $C \in \mathcal{C}$.
- Decrypt. This algorithm takes as inputs **params**, $C \in \mathcal{C}$, and the private keys d_A and t_A, and returns a message $m \in \mathcal{M}$ or a message $\perp$ indicating a decryption failure.

In the above definition, algorithms Extract and Publish can be invoked in either order. There are three private keys in the system. The **master-key** is known to the PKG; $PrvkeyL$ is known to the PKG and the entity, and $PrvKeyR$ is kept secret by the entity itself. Corresponding to the threat of compromising of these keys (note that, the exposure of the **master-key** immediately compromises every $PrvKeyL$), there are two types of adversary. Now we define two games to formalize the adaptive chosen-ciphertext attack secure CL-PKE.

A Type-I adversary which does not know the **master-key**, takes part in the following game (Game 1) with a challenger.

- Setup. The challenger takes a security parameter k and runs the Setup algorithm. It gives the adversary the resulting system parameters **params**. It keeps the **master-key** to itself.
- Phase 1. The adversary issues queries $q_1, \ldots, q_n$ of one of follows:

  - Extraction query on ID_i (of $PrvKeyL$). The challenger responds by running algorithm Extract to generate the private key d_{ID_i} and passes it to the adversary.
  - Publish query on ID_i. The challenger runs algorithm Publish, passes N_{ID_i} to the adversary and maintains a key pair list to store the generated key pairs $\langle N_{ID_i}, t_{ID_i} \rangle$.
  - Replace query on ID_i with the new public key N'_{ID_i}. The challenger records the updated public key N'_{ID_i} for ID_i.
  - Get $PrvKeyR$ query on ID_i. If the public key of ID_i has not been replaced, the challenger responds with the corresponding t_{ID_i}, otherwise, it aborts the game.
  - Decryption query on $\langle ID_i, C_i, N_i \rangle$. The challenger decrypts the ciphertext by finding d_{ID_i} (through running Extract if necessary) and t_{ID_i} (in the public/private key pair list). If t_{ID_i} cannot be found in the key pair list, the challenger outputs $\perp$.
- Challenge. Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $m_0, m_1 \in \mathcal{M}$, an identity ID_{ch} and the public key N_{ch} on which it wishes to be challenged. The only constraint is that ID_{ch} did not appear in any Extraction query in Phase 1 (so, N_{ch} could have been replaced). The challenger picks a random bit $b \in \{0, 1\}$ and sets C^*=Encrypt(params, ID_{ch}, m_b, N_{ch}). It sends C^* as the challenge to the adversary.
- Phase 2. The adversary issues more queries $q_{n+1}, \ldots, q_m$ where query q_i is one of:
  - Extraction query on ID_i where $ID_i \neq ID_{ch}$. The challenger responds as in Phase 1.
  - Publish query on ID_i. The challenger responds as in Phase 1.
  - Replace query on ID_i with the new public key N'_{ID_i}. The challenger responds as in Phase 1.
  - Get $PrvKeyR$ query on ID_i. The challenger responds as in Phase 1.
  - Decryption query on $\langle ID_i, C_i, N_i \rangle \neq \langle ID_{ch}, C^*, N_{ch} \rangle$. The challenger responds as in Phase 1.
- Guess. Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b' = b$.

A Type-II adversary which has the **master-key** (so, it knows every entity's private key $PrivKeyL$) takes part in the following game (Game 2) with a challenger. As this game simulates the scenario of using the traditional PKE and in the adaptive chosen-ciphertext attack game of this setting in the literature [15][4], the adversary does not adaptively corrupt (recovering the private keys of) entities of an entity set, here, we define the game in the same way to disallow the corrupt operation.

- The challenger takes a security parameter k and runs the Setup algorithm. It gives the adversary both the resulting system parameters **params** and the **master-key**.
- The adversary chooses the victim entity with identity ID_{ch}.

- Phase 1. The adversary issues queries $q_1, \ldots, q_n$ of one of follows:
  - Publish key query on ID_i. The challenger runs algorithm Publish, passes N_{ID_i} to the adversary and maintains a key pair list to store the generated key pairs $\langle N_{ID_i}, t_{ID_i} \rangle$.
  - Decryption query on $\langle ID_{ch}, C_i, N_{ch} \rangle$. The challenger responds in the same way as in the Decryption query in Phase 1 of Game 1.
- Challenge. Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $m_0, m_1 \in \mathcal{M}$, on which it wishes to be challenged. The challenger picks a random bit $b \in \{0, 1\}$ and sets C^*=Encrypt(params, ID_{ch}, m_b, N_{ch}). It sends C^* as the challenge to the adversary.
- Phase 2. The adversary issues more queries $q_{n+1}, \ldots, q_m$ where query q_i is one of:
  - Publish key query on ID_i. The challenger responds as in Phase 1.
  - Decryption query on $\langle ID_{ch}, C_i, N_{ch} \rangle$ where $C_i \neq C^*$. The challenger responds as in Phase 1.
- Guess. Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b' = b$.

We refer to these two types of adversary as IND-CCA Type-I (and Type-II) adversary. The advantage of an IND-CCA Type-I (Type-II) adversary $\mathcal{A}$ against the scheme $\mathcal{E}$ is the function of security parameter k: $Adv^{I}_{\mathcal{E},\mathcal{A}}(k) = |Pr[b' = b] - 1/2|$ ($Adv^{II}_{\mathcal{E},\mathcal{A}}(k) = |Pr[b' = b] - 1/2|$).

Definition 1 *A CL-PKE scheme $\mathcal{E}$ is IND-CCA secure if for any IND-CCA Type-I (and Type-II) adversary, $Adv^{I}_{\mathcal{E},\mathcal{A}}(k)$ (and $Adv^{II}_{\mathcal{E},\mathcal{A}}(k)$) is negligible.*

The security definition here, which is more like the security definition without certificate updating in [14], differs from [3] in two ways. First, our definition of the Type-II adversary which follows the standard IND-CCA2 definition in the literature, is more conservative than the one in [3]. For more of (and the difficulty to construct) the adaptive-corrupting chosen-ciphertext attack secure schemes, please see [8][9]. Second, in the model of [3], it is required that even if N_i is a replaced public key (by the adversary) of entity ID_i and is not in the key pair list maintained by the challenger, the challenger still needs to answer the Decrypt query correctly somehow with great probability. It appears that if the scheme $\mathcal{E}$ is secure against any IND-CCA Type-II adversary defined in [3], the challenger in Game 1 could not be able to decrypt the query without knowing the private key corresponding to the used public key and at the same time without the help of some extra facility. Otherwise, we can use the challenger in Game 1 as the adversary to win Game 2 defined in [3]. Note that in the random oracle model, the challenger in the games indeed has an extra advantage over the adversary if a random oracle is used, i.e., it has the full access to the complete input/output list maintained by the random oracle. Hence, it is possible that the challenger can answer such peculiar type of decryption query while the adversary cannot win Game 2. In the formulation here, we do not require the challenger to answer such decryption query successfully. We argue that the formulation can simulate the

chosen-ciphertext attack in the practice. If an adversary replaces B's public key with N'_B generated by itself and A encrypts a message with N'_B, we should not expect that B could decrypt the message successfully, when the adversary uses B as a decryption oracle. Otherwise, the public key N'_B must be used in trivial means in algorithm Encrypt. On the other hand, if this is true, the adversary can easily win Game 2. The formulation also covers the malicious behavior that the adversary replaces B's public key with C's and after getting the ciphertext encrypted for B with C's public key, asks B and C to cooperate to decrypt it somehow. In this case, the challenger will answer the decryption query correctly. In fact, this simulation maybe is still unnecessarily strong.

A secure CL-PKE scheme achieves two important properties that differ a CL-PKE from either an IBE or a traditional PKE. First, the public key of an entity can be loosely (no need of security measures) bound with the identity of the entity because a CL-PKE is secure against Type-I adversaries. This is a big advantage over the traditional PKE. Second, a secure CL-PKE achieves the **master-key** forward secrecy against Type-II adversaries (i.e., key-escrow free) which is not achievable in the IBE following Shamir's framework [20], while needed and realized in the traditional PKE (the compromise of the signing key of a Certificate Authority (CA) does not pose the threat on existing encrypted messages and a CA cannot decrypt a message encrypted for an entity of which only the public key is known by the CA).

2.2 Bilinear Groups

Here we briefly review some facts about bilinear groups and pairings used in the schemes in this paper.

Definition 2 *A pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with two cyclic group $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, which has the following properties [6]:*

1. *Bilinear: $\hat{e}(sP, tR) = \hat{e}(P, R)^{st}$ for all $P, R \in \mathbb{G}_1$ and $s, t \in \mathbb{Z}_q^*$.*
2. *Non-degenerate: For a given point $Q \in \mathbb{G}_1$, $\hat{e}(Q, R) = 1_{\mathbb{G}_2}$ for all $R \in \mathbb{G}_1$ if and only if $Q = 1_{\mathbb{G}_1}$.*
3. *Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.*

The modified Weil and Tate pairings on elliptic curves can be used to build such bilinear maps [23].

Assumption 1 Bilinear Diffie-Hellman Assumption (BDH) *[6] Let $\mathcal{G}$ be a parameter generator which with system parameters 1^k as input generates two cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order q, a generator $P \in \mathbb{G}_1^*$ and a bilinear map $\hat{e}$. We define the advantage of an algorithm $\mathcal{A}$ in solving the problem (given $\langle P, aP, bP, cP \rangle$, to compute $\hat{e}(P, P)^{abc}$) by:*

$$Adv_{\mathcal{G},\mathcal{A}}(k) = Pr[\, \mathcal{A}(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P, P)^{abc} | \\ \langle q, \mathbb{G}_1, \mathbb{G}_2, P, \hat{e} \rangle \leftarrow \mathcal{G}(1^k), P \in \mathbb{G}_1^*, a, b, c \xleftarrow{R} \mathbb{Z}_q^*].$$

For any randomized polynomial time (in k) algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{G},\mathcal{A}}(k)$ is negligible.

Note that the BDH assumption implies the following Computational Diffie-Hellam (CDH) assumption in group $\mathbb{G}_1$.

Assumption 2 Computational Diffie-Hellman Assumption (CDH) *Let $\mathcal{G}$ be a parameter generator which with system parameters 1^k as input generates a group $\mathbb{G}_1$ of prime order q and a generator $P \in \mathbb{G}_1^*$. We define the advantage of an algorithm $\mathcal{A}$ in solving the problem (given $\langle \mathbb{G}_1, P, aP, bP \rangle$, to compute abP) by:*

$$Adv_{\mathcal{G}_1,\mathcal{A}}(k) = Pr[\, \mathcal{A}(q, \mathbb{G}_1, P, aP, bP) = abP | \\ \langle q, \mathbb{G}_1, P, \rangle \leftarrow \mathcal{G}(1^k), P \in \mathbb{G}_1^*, a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q^*].$$

For any randomized polynomial time (in k) algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{G},\mathcal{A}}(k)$ is negligible.

2.3 Existing Schemes

Apart from introducing the notion of CL-PKC, Al-Riyami and Paterson also proposed a concrete scheme which extends Boneh-Franklin's IBE based on the so called general BDH assumption, i.e., given $(\mathbb{G}_1, \mathbb{G}_2, P, aP, bP, cP)$ such that $a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$, it is hard to output a pair $(Q \in \mathbb{G}_1^*, \hat{e}(P, Q)^{abc})$. A certificateless signature and a hierarchical encryption system are constructed as well [3].

It is not difficult to demonstrate that the scheme proposed by Gentry in [14] is also a CL-PKE. Gentry's scheme is different from AP's in a few ways. First, an entity with identity ID_A publishes $info_A$ including a single element $t_A P \in \mathbb{G}_1^*$ ($t_A \in \mathbb{Z}_q^*$ is the private key $PrvKeyR$) and some other information (naturally, the identity ID_A). Second, algorithm Extract takes $info_A$, the **master-key** s and **params** as input and returns $d_A = sH_1(P_{pub} \| info_A)$ ($a \| b$ denotes the concatenation of two strings a and b) as $PrvKeyL$ of A. Algorithms Encrypt and Decrypt work differently from AP's as well.

An authenticated CL-PKE is intended in [18] to improve the performance of AP's scheme. However, the scheme is not secure against the Type-I adversary. It is easy to check that if the adversary randomly chooses $x_1, x_2 \in \mathbb{Z}_q^*$ and publishes $\langle x_1 P, x_2 P \rangle$ as $\langle X_B, Y_B \rangle$, the adversary can recover the decryption immediately by computing $T = x_1 x_A P$ and $\hat{e}(d_A, Y_B)^r = \hat{e}(rQ_A, x_2 P_{pub})$.

3 The New CL-PKE

Pairing is a very heavy operation compared with the point scalar, exponentiation and hash operations. In the above two CL-PKE's, AP's scheme needs three (resp. one) pairings in algorithm Encrypt (resp. Decrypt), while Gentry's scheme needs two (resp. one) pairings in Encrypt (resp. Decrypt). Here we present another construction which is more efficient in encryption. Just as intended by the CL-PKE (to combine the advantage of both the traditional PKE and the IBE), our

scheme are exactly the integration, using a hash function as a hedge (borrowed from [21]), of two algorithms with one of each type, i.e., Boneh-Franklin's IBE [6] and a variant of ElGamal's Diffie-Hellman encryption scheme [12][1] strengthened using Fujisaki-Okamoto's transform [13]. We believe that this method can be generalized.

Setup. Given a security parameter k, the parameter generator follows the steps.

1. Generate two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Pick a random generator $P \in \mathbb{G}_1^*$.
2. Pick a random $s \in \mathbb{Z}_q^*$ and compute $P_{pub} = sP$.
3. Pick four cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 : \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$ for some integer $n > 0$.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$. The system parameters are $\mathbf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$. s is the **master-key** of the system.

Extract. Given a string $ID_A \in \{0,1\}^*$, **params** and the **master-key**, the algorithm computes $Q_A = H_1(ID_A) \in \mathbb{G}_1^*$, $d_A = sQ_A$ and returns d_A.

Publish. Given **params**, an entity A selects a random $t_A \in \mathbb{Z}_q^*$ and computes $N_A = t_A P$. The entity can ask the PKG to publish N_A or publishes it by itself or via any directory service as its public key (or nickname).

Encrypt. Given a plaintext $m \in \mathcal{M}$, the identity ID_A of entity A, the system parameters **params** and the public key N_A of the entity, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m)$.
2. Compute $Q_A = H_1(ID_A)$, $g^r = \hat{e}(P_{pub}, Q_A)^r$ and $f = rN_A$.
3. Set the ciphertext to $C = \langle rP, \sigma \oplus H_2(rP, g^r, f), m \oplus H_4(\sigma) \rangle$.

Decrypt. Given a ciphertext $\langle U, V, W \rangle \in \mathcal{C}$, the private keys d_A, t_A and **params**, follow the steps:

1. Compute $g' = \hat{e}(U, d_A)$, $f' = t_A U$ and $\sigma' = V \oplus H_2(U, g', f')$
2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m')$.
3. If $U \neq r'P$, output $\perp$, else return m' as the plaintext.

The consistency of the scheme can be easily verified. The security of the scheme in the random oracle model can be proved rather straightforwardly based on the existing work. The scheme's property against Type-I adversaries follows directly from the security proof of Boneh-Franklin's IBE. The property against Type-II adversaries fairly straightly follows from the work in [13] based on the CDH assumption in $\mathbb{G}_1$ implied by the BDH assumption. The use of rP in H_2 is

suggested in [22][10] which makes use of the Gap Diffie-Hellman[1] assumption in $\mathbb{G}_1$ in the security proof to achieve non-malleability[2] [4] and a tighter reduction. We skip the tedious details here.

Theorem 1 *The above scheme is an IND-CCA-secure CL-PKE provided that H_1, H_2, H_3, and H_4 are random oracles and the BDH assumption is sound.*

4 A CL-Auth-PKE

The new CL-PKE scheme can be simply extended to an authenticated encryption (Auth-PKE)(not signcryption) [17][2][24]. The formulation of identity-based Auth-PKE (with message privacy and authentication property) can be found in [17] which can be extended to the certificatless setting with considering the extra public key. However, the formulation and construction in [17] (and the *outsider security* in [2]) does not achieve the forward secrecy of sender's private keys (we shorthand as *forward secrecy*), i.e., if the sender's private keys are compromised, any message encrypted with these keys, could be recovered. We think this is an unattractive property in the practice. Further analysis shows that there are two subcases. Case 1, the adversary has the interest to recover the decryption of those ciphertexts auth-encrypted before it compromises the sender's private keys. Case 2, although the private keys were exposed, an entity continues to use these keys to auth-encrypt messages (maybe the attack is so smart that the adversary cannot notice the fact that its keys are leaked). The adversary wants to decrypt the auth-encrypted messages after the compromission of keys as well. In Case 1, the scheme only needs to achieve forward secrecy, while in Case 2, the scheme has to guarantee both forward and (we call it) *backward secrecy*. Moreover, as in Case 2, the compromised entity is still active, it could be used as a decryption oracle. Hence, in Case 2, we should design a scheme against adaptive chosen ciphertext attacks, while in Case 1, a chosen plaintext attack Auth-PKE (IND-CPA-Auth-PKE) that is secure on the prerequisite of the secrecy of receiver's private keys is enough (for well-known reason, one-way encryption [13] is inadequate).

The *forward secrecy* of an Auth-PKE against Type-I adversaries is defined by the following sender-key-known CPA game.

- Setup. The challenger takes a security parameter k and runs the Setup algorithm. It gives the adversary the resulting system parameters **params**. It keeps the **master-key** to itself.
- Query phase. The adversary issues the following queries.

[1] The Computational DH still appears hard, even in the presence of a Decisional DH oracle [19].

[2] Although proved in [4] that the non-malleability and IND-CCA2 imply each other, to prove that the scheme without using rP in H_2 is secure against Type-II adversaries, we need a prerequisite that rP is uniquely represented in the ciphertext space [22].

  - Extraction query on ID_s (of $PrvKeyL$). The challenger responds by running algorithm Extract to generate the private key d_{ID_s} and passes it to the adversary.
  - Publish query on ID_s and ID_r. The challenger runs algorithm Publish, passes N_{ID_s} and N_{ID_r} to the adversary and maintains a key pair list to store the generated key pairs.
  - Get $PrvKeyR$ query on ID_s and ID_r. The challenger responds with t_{ID_s} and t_{ID_r} which correspond to the public key N_{ID_s} and N_{ID_r} respectively.
- Challenge. The adversary outputs two equal length plaintexts $m_0, m_1 \in \mathcal{M}$, and the public key N'_{ID_r} on which it wishes to be challenged. The challenger picks a random bit $b \in \{0,1\}$ and sets C^*=Auth-Encrypt(params, $ID_s, d_{ID_s}, t_{ID_s}, N_{ID_s}, m_b, ID_r, N'_{ID_r}$), i.e., m_b is encrypted for the receiver with identity ID_r and public key N'_{ID_r} by the sender with identity ID_r and public/private key pair N_{ID_s}/t_{ID_s}. It sends C^* as the challenge to the adversary. Note that N'_{ID_r} could be different from N_{ID_r}, while N_{ID_s} has to be the one published by the challenger.
- Guess. Finally, the adversary outputs a guess $b' \in \{0,1\}$ and wins the game if $b' = b$.

The advantage of a sender-key-known IND-CPA Type-I adversary $\mathcal{A}$ against the Auth-PKE scheme $\mathcal{E}$ is the function of security parameter k: $Adv^{I-CPA}_{\mathcal{E},\mathcal{A}}(k) = |Pr[b' = b] - 1/2|$. Note that, the above game does not allow the adversary to adaptively choose the sender or receiver. Similarly, we can define another game to define the *forward secrecy* of an Auth-PKE against Type-II adversaries. So, we say that a CL-Auth-PKE is secure only if it achieves message privacy, message authentication and forward secrecy of sender's private keys.

Here, we construct a CL-Auth-PKE scheme achieving the *forward secrecy* (in fact, the *backward secrecy* as well, see the argument in Appendix A).
Setup. Same as the algorithm Setup in the CL-PKE scheme, except that the hash function H_2 is defined as $H_2 : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^n$.

Extract and **Publish** work in the same way as in the CL-PKE scheme.

Auth-Encrypt. Given a plaintext $m \in \mathcal{M}$, identity ID_A, public key N_A, private keys d_A, t_A of sender A, identity ID_B and public key N_B of receiver B and **params**, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m)$.
2. Compute $Q_A = H_1(ID_A)$, $Q_B = H_1(ID_B)$, $g^r = \hat{e}(d_A, Q_B)^r$ and $f = rt_A N_B$.
3. Set the ciphertext to $C = \langle rN_A, rQ_A, \sigma \oplus H_2(rN_A, rQ_A, g^r, f), m \oplus H_4(\sigma)\rangle$.

Auth-Decrypt. Given a ciphertext $\langle T, U, V, W\rangle \in \mathcal{C}$, the private keys d_B, t_B and identity ID_B of receiver B, the public key N_A and identity ID_A of sender A, and **params**, follow the steps:

1. Compute $g' = \hat{e}(U, d_B)$, $f' = t_B T$ and $\sigma' = V \oplus H_2(T, U, g', f')$

2. Compute $m' = W \oplus H_4(\sigma')$, $r' = H_3(\sigma', m')$ and $Q_A = H_1(ID_A)$.
3. If $U \neq r'Q_A$ or $T \neq r'N_A$, output $\perp$, else return m' as the plaintext.

Adopting the similar method in the proof of Theorem 4.1 in [6], we can prove that the above scheme achieves the forward secrecy against Type-I adversaries. Please see Appendix A for part of the proof. Using the similar way to prove that ElGamal's DH encryption strengthened by Fujisaki-Okamoto's transform is IND-CCA secure, we can prove that the scheme is forward secure against Type-II adversaries. Note that the scheme can be sender-anonymous, i.e., no second party (apart from the intended receiver) can recover the sender's identity merely from the ciphertext. In this case, the sender's identity is part of the message being auth-encrypted.

5 Remarks on CL-PKE's

First, we evaluate the complexity of three schemes. The schemes have four major operations, i.e., Pairing, Scalar, Exponentiation and Hash. Pairing is the heaviest one which involves the point scalar as one of the basic operations, even if many techniques can be applied on pairing operation to dramatically improve the performance [7]. Note that some point scalar operations in $\mathbb{G}_1$ and exponentiations in $\mathbb{G}_2$ can be converted to each other and the performance difference of these two operations heavily depends on the specific implementation. We count these operations as they are presented in the schemes. Without considering the pre-computation, the complexity of the schemes is listed in Table 1. Note that three schemes have the same ciphertext length (except the Auth-Encryption scheme) and some operations need to be done only once in the whole life circle of the public/private key pair, e.g., the scalar $t_A d_A$ in Decrypt of AP's scheme and $Q_A^1 = H_1(info_A)$ in Decrypt of Gentry's scheme. We do not count these operations.

	Encrypt				Decrypt				pubkey len
	P	S	E	H	P	S	E	H	
Gentry's scheme	2	1	1	5	1	1	0	3	1*
AP's scheme	3	1	1	4	1	1	0	3	2
New scheme	**1**	**2**	**1**	**4**	**1**	**2**	**0**	**3**	**1**
Auth-Encryption scheme	1	3	1	4	1	3	0	3	1

Table 1. The Complexity of CL-PKE's

We can find that the new scheme is fastest in Encrypt (which is more significant in a hierarchical system such as [16][3][14]), while relatively slower than other two schemes in Decrypt. AP's scheme needs to publish two elements of $\mathbb{G}_1$, while the new scheme needs only one. This would save the bandwidth in some situations, e.g., if the scheme is used in a key exchange protocol which needs the

sender to piggy-send its public key in a message. Note that in Gentry's scheme, $info_A$ of entity A could also include only a single element in $\mathbb{G}_1$ and its identity. The new scheme does have a disadvantage, i.e, there are two private keys managed by each entity, while in other two schemes, two private keys can be combined as one, so to save storage. We also note that if the result of Step 1 in algorithm Encrypt of AP's scheme can be reused, AP's scheme becomes most efficient on computation.

Gentry's scheme works more like a traditional PKE. In the scheme, an entity first generates its public/private key pair and then asks the PKG to issue a certificate to bind its identity with its public key. The disadvantage of this method is that the entity cannot freely change its public key without interacting with the PKG. On the other hand, we will see later that this method has two important advantages. A good aspect of AP's scheme is that it can be easily extended to other certificateless protocols including signature. Note that the new scheme can be easily modified to integrate AP's solution. In the modified scheme, entity A publishes its public key $N_A = (X_A, Y_A) = (t_A P, t_A sP)$ as in AP's. In algorithm Encrypt, f is computed by $f = rX_A$ and then certificateless signature scheme just works as the one in [3]. All the schemes can be converted to a KEM (similar to ECIES-KEM) to be used in a hybrid encryption [22][10] (see Appendix B).

Finally, we give two comments on the CL-PKC in general. (1) In the CL-PKC system, entities have to trust that the PKG will not launch an active attack to replace an entity's public key with its own choice. Although in the traditional PKC, entities have to trust the CA as well, there is a fundamental difference between these two systems. If a CA launch such attack, it will leave a trace (a valid certificate) to face the legal penalty. While, in the CL-PKC, although suggested in [3] and implemented in [14] that an entity's public key and the identity are bound together to generated $PrvKeyL$ (hence, only the PKG can generate this key corresponding to the public key. *Note that the new scheme can play this trick as well*), only when the $PrvKeyL$ is used in an undeniable operation such as signing a message, the PKG's misbehavior could be traced (one advantage of Gentry's $PrvKeyL$ generation method). This may be not good enough in many settings. (2) The public key revocation is still a great challenge. If the two private keys $PrvKeyL$ and $PrvKeyR$ are both compromised, the entity has to publish a revocation message to prevent others from using the corresponding public key to encrypt messages. So, a public key revocation list has to be maintained *securely*, just as the certificate revocation list (CRL) in the traditional PKC. And if the entity wants to keep its identity, then the system should generate $PrvKeyL$ in the same fashion as Gentry's, i.e., $PrvKeyL_A = sH_1(ID_A \| N_A)$ (another advantage of Gentry's $PrvKeyL$ generation method). We seem to come back to the starting point, to face the scalability issue. Some other solutions such as key evolvement scheme [14], intrusion-resilient encryption [11], mediated encryption [5], etc, have been attempted. None of them can fully solves the problem.

References

1. M. Abdalla, M. Bellare and P. Rogaway, "DHIES: An encryption scheme based on the Diffie-Hellman Problem," extended abstract, entitled The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES, in Topics in Cryptology - CT-RSA 2001, LNCS Vol. 2020, 2001.
2. J. H. An, Y. Dodis and T. Rabin, "On the security of joint signature and encryption," In L. Knudsen, editor, Advances in Cryptology-Eurocrypt 2002, LNCS Vol. 2332, 2002.
3. S. S. Al-Riyami and K. G. Paterson, "Certificateless Public Key Cryptography," Advances in Cryptology-Asiacrypt 2003, LNCS 2894, 2003. See also Cryptology ePrint Archive, Report 2003/126.
4. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in Advances in Cryptology CRYPTO 1998, LNCS Vol. 1462, 1998.
5. D. Boneh, X. Ding, G. Tsudik and C. Wong, "A Method for Fast Revocation of Public Key Certificates and Security Capabilities," Proceedings of the 10th USENIX Security Symposium, USENIX, 2001.
6. D. Boneh and M. Franklin, "Identity Based Encryption from The Weil Pairing," extended abstract in Advances in Cryptology-Crypto 2001, LNCS Vol. 2139, 2001.
7. P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," Advances in Cryptology-Crypto 2002, LNCS Vol. 2442, pp. 354-368, 2002. See also Cryptology ePrint Archive, Report 2002/008.
8. R. Canetti, U. Feige, O. Goldreich and M. Naor, "Adaptively Secure Computation," 28th ACM Symposium on Theory of Computing (STOC), ACM, pp. 639-648, 1996. Full version in MIT-LCS-TR #682, 1996.
9. R. Canetti, S. Halevi and J. Katz, "Adaptively-Secure, Non-Interactive Public-Key Encryption," extended abstract at TCC 2005. See also Cryptology ePrint Archive, Report 2004/317.
10. R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," SIAM Journal of Computing 33:167-226, 2003.
11. Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung, "Intrusion-Resilient Public-Key Encryption," Topics in Cryptology - CT-RSA 2003, LNCS 2612, pp. 19-32, 2003.
12. T. ElGamal, "A public key cryptosystem and signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, 31:469-472, 1985.
13. E. Fujisaki and T. Okamotom, "Secure Integration of Asymmetric and Symmetric Encryption Schemes," Advances in Cryptology - CRYPTO 1999 Proceedings, pp.535-554, Springer-Verlag, 1999.
14. C. Gentry, "Certificate-Based Encryption and the Certificate Revocation Problem," Cryptology ePrint Archive, 2003/183.
15. S. Goldwasser and S. Micali, "Probabilistic encryption," Journal of Computer and System Sciences Vol. 28, 270-299, 1984.
16. C. Gentry and A. Silverberg, "Hierarchical ID-Based Cryptography," in Proceedings of Asiacrypt 2002, LNCS Vol. 2501, 2002.
17. B. Lynn, "Authenticated Identity-Based Encryption," Cryptology ePrint Archive, Report 2002/072.
18. Y.-R. Lee and H.-S. Lee, "An Authenticated Certificateless Public Key Encryption Scheme," Cryptology ePrint Archive, 2004/150.

19. T. Okamoto and D. Pointcheval, "The gap-problems: a new class of problems for the security of cryptographic schemes," In Proc. 2001 International Workshop on Practice and Theory in Public Key Cryptography (PKC 2001), 2001.
20. A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," in Advances in Cryptology-Crypto 1984, LNCS Vol. 196, 1984.
21. V. Shoup, "Using hash functions as a hedge against chosen ciphertext attack," in Proc. Eurocrypt 2000.
22. V. Shoup, "A Proposal for an ISO Standard for Public Key Encryption," 2001.
23. E. Verheul, "Evidence that XTR is more secure than supersingular elliptic curve cryptosystems," Advances in Cryptology–Eurocrypt 2001, LNCS Vol. 2045, pp. 195-210, 2001.
24. Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption)," In B. Kaliski, editor, Advances in Cryptology-Crypto 1997, LNCS 1294, Vol. 1294, 1997.

Appendix A

We can follow the method in [6] to prove that the CL-Auth-PKE achieves the forward secrecy against Type-I adversaries. Here we present only the proof of a conclusion similar to Lemma 4.3 in [6]. Readers can extend the result to a full proof as in [6].

First, we define a **BasicAuthPub** scheme consisting of following algorithms.
Keygen. Given a security parameter k, the parameter generator follows the steps.

1. Generate two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Pick a random generator $P \in \mathbb{G}_1^*$.
2. Pick a random $s \in \mathbb{Z}_q^*$ and compute $P_{pub} = sP$. Pick two random elements $Q_A, Q_B \in \mathbb{G}_1^*$. Pick two elements $t_A, t_B \in \mathbb{Z}_q^*$.
3. Pick a cryptographic hash function $H_2 : \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \{0,1\}^n$ for some integer $n > 0$.
4. The public key is $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, n, \hat{e}, P, P_{pub}, Q_A, t_AP, Q_B, t_BP, H_2 \rangle$. The private keys are $t_A, d_A = sQ_A, t_B, d_B = sQ_B$.

Encrypt. Given a plaintext $m \in \{0,1\}^n$, K_{pub} and the private keys d_A, t_A, the following steps are performed.

1. Pick a random $r \in \mathbb{Z}_q^*$ and compute $g^r = \hat{e}(d_A, Q_B)^r$ and $f = rt_At_BP$.
2. Set the ciphertext to $C = \langle rt_AP, rQ_A, m \oplus H_2(rt_AP, rQ_A, g^r, f) \rangle$.

Decrypt. Given a ciphertext $\langle T, U, V \rangle$ encrypted using the public key K_{pub}, and private keys d_B, t_B, follow the step:

1. Compute $g' = \hat{e}(U, d_B)$, $f' = t_BT$ and return $m = V \oplus H_2(T, U, g', f')$ as the plaintext.

Lemma 1 *Let H_2 be a random. Let $\mathcal{A}$ be an sender-key-known IND-CPA adversary that has advantage $\epsilon(k)$ against* **BasicAuthPub**. *Suppose $\mathcal{A}$ makes a total of $q_{H_2} > 0$ queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem with advantage at least $2\epsilon(k)/q_{H_2}$ and a running time $O(time(\mathcal{A}))$.*

Proof: For easy to prove the lemma, we extend the BDH assumption to the following assumption (EBDH). Given $\langle q, \mathbb{G}_1, \mathbb{G}_2, R, aR, bR, cR, vR, vaR, vcR, t_1, t_1vR, t_1vcR, t_2, t_2vR\rangle$ such that $R \in_R \mathbb{G}_1^*$ and $a, b, c, v, t_1, t_2 \in_R \mathbb{Z}_q^*$, it is hard to compute $\hat{e}(R,R)^{abc}$. Readers can easily verify that the BDH and EBDH assumptions imply each other with a trivial reduction.

Algorithm $\mathcal{B}$ given a random EBDH instance interacts with $\mathcal{A}$ in the following way (using $\mathcal{A}$ as a subroutine).

Setup. Algorithm $\mathcal{B}$ creates the **BasicAuthPub** public key K_{pub} in the following way. $\mathcal{B}$ sets K_{pub} as $\langle q, \mathbb{G}_1, \mathbb{G}_2, n, \hat{e}, vR, vaR, R, t_1vR, bR, t_2vR, H_2\rangle$, i.e., $\mathcal{B}$ sets $P = vR$, $s = a$ (which $\mathcal{B}$ does not known), $P_{pub} = vaR$, $Q_A = R$, $t_A = t_1$, $t_AP = t_1vR$, $Q_B = bR$, $t_B = t_2$, $t_BP = t_2vR$ and H_2 is a random oracle controlled by $\mathcal{B}$. Note that by definition, the private keys are $d_A = sQ_A = aR$, t_A, $d_B = sQ_B = abR$ (which $\mathcal{B}$ does not know) and t_B.

H_2-queries (T_i, U_i, X_i, Y_i). At any time algorithm $\mathcal{A}$ can issues queries to the random oracle H_2. To response these queries $\mathcal{B}$ maintains a list of tuples called H_2^{list}. Each entry in the list is a tuple of the form $\langle T_i, U_i, X_i, Y_i, H_i\rangle$. To response a query on (T_i, U_i, X_i, Y_i), $\mathcal{B}$ does the following operations:

1. If (T_i, U_i, X_i, Y_i) is on the list in a tuple $\langle T_i, U_i, X_i, Y_i, H_i\rangle$, then $\mathcal{B}$ responds with $H_2(T_i, U_i, X_i, Y_i) = H_i$.
2. Otherwise, $\mathcal{B}$ randomly chooses a string $H_i \in \{0,1\}^n$ and adds the tuple $\langle T_i, U_i, X_i, Y_i, H_i\rangle$ to the list. It responds to $\mathcal{A}$ with $H_2(T_i, U_i, X_i, Y_i) = H_i$.

Query phase.

- Extraction query on ID_s. $\mathcal{B}$ responds with aR.
- Publish query on ID_s and ID_r. $\mathcal{B}$ responds with t_AvR and t_BvR.
- Get $PrvKeyR$ query on ID_s and ID_r. $\mathcal{B}$ responds with t_A and t_B.

Challenge phase. Algorithm $\mathcal{A}$ outputs two messages m_0, m_1 and N_B' on which it wants to be challenged. $\mathcal{B}$ chooses a random string $V \in \{0,1\}^n$ and defines $C_{ch} = \langle t_AvcR, cR, V\rangle = \langle T, U, V\rangle$. $\mathcal{B}$ gives C_{ch} as the challenge to $\mathcal{A}$. Note that, by definition, $U = rQ_A = cR$ (which implies $r = c$ because $Q_A = R$), $T = rt_AP = rt_AvR = t_AvcR$ and the decryption of C is $V \oplus H_2(T, U, \hat{e}(U, d_B), t_BT)$ where $\hat{e}(U, d_B) = \hat{e}(cR, abR) = \hat{e}(R,R)^{abc}$.

Guess. Algorithm $\mathcal{A}$ outputs its guess $c' \in \{0,1\}$. At this point $\mathcal{B}$ picks a random tuple $\langle T_i, U_i, X_i, Y_i, H_i\rangle$ from the list $\mathcal{H}_2^{list}$ and outputs X_i as the solution to the EBDH instance.

Following the same argument in the proof of Lemma 4.3 in [6], we have that $\mathcal{B}$ outputs the correct answer to the EBDH instance with probability at least $2\epsilon(k)/q_{H_2}$. In fact in the Guess phase, $\mathcal{B}$ could randomly choose a tuple from a set S which includes the tuples whose $T_j = T$ and $U_j = U$ on list H_2^{list}. Then, because of the randomness of cR, a tighter reduction could be obtained.

□

Lemma 1 shows that the **BasicAuthPub** already achieves the forward secrecy again Type-I adversaries. By applying Fujisaki-Okamoto's transform, the full scheme is secure against sender-key-known CCA adversaries. Note that, this simple reduction does not guarantee the security against the adaptively corrupting adversaries implied in Boneh-Franklin's proof [6].

Appendix B

A ECIES-KEM-similar hybrid encryption [1][22] consists of following algorithms.
Setup. Given a security parameter k, the parameter generator follows the steps.

1. Generate two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Pick a random generator $P \in \mathbb{G}_1^*$.
2. Pick a random $s \in \mathbb{Z}_q^*$ and compute $P_{pub} = sP$.
3. Pick two cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $KDF : \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^n \times \{0,1\}^l$ for some integers $n, l > 0$.
4. Pick a symmetric encryption algorithm $ENC_{k_1}(\cdot)$ which uses n-bit k_1 as the key. Pick a keyed-hash function $MAC_{k_2} : \{0,1\}^* \to \{0,1\}^t$ for some integer $t > 0$, which uses l-bit k_2 as the key.

The message space is $\mathcal{M} = \{0,1\}^*$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^* \times \{0,1\}^t$. The system parameters are $\mathbf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, KDF, ENC_{(\cdot)}, MAC_{(\cdot)} \rangle$. s is the **master-key** of the system.

Publish. Given **params**, an entity A selects a random $t_A \in \mathbb{Z}_q^*$ and computes $N_A = t_A P$. The entity can ask the PKG to publish N_A or publishes it by itself or via any directory service as its public key.

Extract. Given a string $ID_A \in \{0,1\}^*$, the public key N_A generated in Publish, **params** and the **master-key**, the algorithm computes $Q_A = H_1(ID_A \| N_A) \in \mathbb{G}_1^*$, $d_A = sQ_A$ and returns d_A.

Encrypt. Given a plaintext $m \in \mathcal{M}$, the identity ID_A of entity A, the system parameters **params** and the public key N_A of the entity, the following steps are performed.

1. Pick a random $r \in \{0,1\}^n$ and compute $Q_A = H_1(ID_A \| N_A)$, $g^r = \hat{e}(P_{pub}, Q_A)^r$ and $f = rN_A$.
2. Compute $\langle k_1, k_2 \rangle = KDF(rP, g^r, f)$;
3. Compute $c = ENC_{k_1}(m)$;
4. Compute $t = MAC_{k_2}(c)$.
5. Set the ciphertext to $C = \langle rP, c, t \rangle$.

Decrypt. Given a ciphertext $\langle U, V, W \rangle \in \mathcal{C}$, the private keys d_A, t_A and **params**, follow the steps:

1. Compute $g' = \hat{e}(U, d_A)$, $f' = t_A U$ and $\langle k_1, k_2 \rangle = KDF(rP, g', f')$,

2. Verify that $W = MAC_{k_2}(V)$. If the equation does not hold, return $\perp$ indicating a decryption failure.
3. Compute $m = ENC^{-1}_{k_1}(V)$ as the plaintext.