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

Last update on April 12, 2005

Abstract. In [3] Al-Riyami and Paterson introduced the notion of "Certificateless Public Key Cryptography" and presented an instantiation. In this paper, we revisit the formulation of certificateless public key encryption and construct a more efficient scheme and then 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 with many challenges in the practice, especially the scalability of the infrastructure. In [29], 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).

The intuition behind the certificateless public key encryption is that 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) is introduced to play 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 [8], the identity of an entity can also serve as a public key and the CL-PKE becomes realistic.

So far, there are two major constructions of certificateless public key encryption (CL-PKE), i.e., the scheme proposed by Gentry in [20] and the system in [3] (we call it AP's scheme). In this paper, after revisiting the formulation of CL-PKE, 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 the existing CL-PKE schemes and a used primitive. The new CL-PKE scheme and a tweaked version are presented in Section 3 and then we extend the scheme to an authenticated encryption. Finally, we remark the CL-PKE schemes on complexity and application.

2 Preliminaries

2.1 Revisiting Certificateless Public Key Encryption

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 message space $\mathcal{M}$, and a description of a ciphertext space $\mathcal{C}$. The system parameters will be publicly known, while the **master-key** will be kept secret by 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 both the PKG and the entity, and $PrvKeyR$ is kept secret by the entity itself. Corresponding to the threat of compromising 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[1] to formalize the adaptive chosen-ciphertext attack secure CL-PKE against these adversaries. Note that our security definition is different from the one in [3].

A Type-I adversary which does not know the **master-key**, takes part in the following game [2] (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. Note that in the game, the list is of polynomial length). 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 $q_{n+1}, \ldots, q_l$ where query q_i is one of:

[1] It is more general to define a single game with three separate stages, i.e., Stage 1 to generate the system parameters and the master key, Stage 2 corresponding to the following Game 1, and Stage 3 for Game 2 (both without the Setup phase). The adversary erases any internal state information before entering Stage 3 and before starting Stage 3, the challenger provides some extra information (i.e., the master key) apart from the system parameters to the adversary. The adversary wins any stage (Stage 2 or Stage 3) to win the game. However, this formulation appears hard to use.

[2] Game 1 follows the IBE formulation [8] which allows the adversary to adaptively corrupt the parties. Canetti el al. formulated a weaker security notion, selective identity IBE [14], which forces the adversary to commit the victim's identity at the beginning of the game. Game 1 can be easily modified to a selective identity game.

  - 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, the adversary does not adaptively corrupt entities from an entity set in the literature [22][6], here we define the game in the way disallowing the corrupt operation as well.

- 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 $q_{n+1}, \ldots, q_l$ 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 of CBE without certificate updating in [20], differs from [3] (and some following-up work [35][36][4]) in two significant 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 specified in [3] (and [35][36][4]). In [3], the Type-II adversary is allowed to adaptively corrupt the entities by an *Extract private key for entity A* query. However, it seems that the existence of such kind of encryption scheme with short public/secret keys at the same time allowing adaptive corruption is arguable.

Canetti et al. concluded that "*no noninteractive encryption scheme that supports encryption of an unbounded number of messages and uses a single, unchanging decryption key can be adaptively secure*" [15]. This negative result was original presented in Nielsen's work [27], which argues that "*the number of possible secret keys must be at the least the number of possible message-sequence*" [15]. While this conclusion was reached under a condition that the encryption scheme is used to simulate a secure channel. In the secure channel model, the adversary cannot see the ciphertexts between communicating parties but only knows the length of the messages. Note that this ciphertext unawareness is crucial in Nielsen's argument. Following this argument, Nielsen presented a scheme that is secure in the random oracle model even when the adaptive corruption is allowed, so to provide a separation between the random oracle model and the standard model [27]. While, in [5] the authors raised a concern of Nielsen's separation, which is also about the ciphertext awareness (to the environment in the Universally Composable model [12]).

So, is Canetti et al.'s negative conclusion about the existence of the adaptively-secure non-interactive public-key encryption founded? In fact, this question can be tracked back to the fourth open question raised in [9]. In particular, does the failure of direct using public key encryptions in secure multiparty computation to replace secure channels indicate a deficiency of public key encryption (Canetti et al. designed a special type of encryption, non-committing encryption, to solve this problem [13]) or does it suggest that zero-knowledge based (simulation-based) definitions of protocol security are not suitable for a general approach of security? We think that the following points are of concern regarding the question. First, in the standard public-key encryption security model [22][6], the adversary has the access to the ciphertexts. This is different from a secure channel formulation. A typical example is that Bellare et al. fixed the plaintext awareness formulation [6] to allow the adversary to eavesdrop communication between parties. Second, note that there is a subtle difference between a private key query and a complete corruption. A complete corruption allows the adversary to reveal an entity's current state, although the private key query is very close to a complete corruption in a multiparty computation model with erasure (note that even in this case, Nielsen's negative result is sustained as well). Third, recent work [34] shows that the identity-based encryption in the full model (allowing the adaptive private key query) without random oracle can be constructed. This seems to be the encouraging news for constructions of normal adaptively-secure non-interactive public-key encryptions. (Note that Canetti et al.'s construction has some constraints [15].)

Here we cannot reach a conclusion, but simply raise this concern. And we think that maybe it is not prudent to simply assume, without further investigation, that the extra adaptive private key extraction ability does not help the adversary to win the game which simulates a standard encryption construction. Hence here we adopt the standard formulation as in [6][22].

One point of the formulation is worth mentioning, which also sidedly backs up our conservatism. In Game 2, the adversary is allowed to choose the challenge identity ID_{ch}. This defines a stronger (or at least equivalent) security notion than a game allows the challenger to choose the ID_{ch}. The intuition follows from the same argument of difficulty to construct adaptively-secure multiparty computation protocols.

Static corruption (allowing the adversary to corrupt some parties after Setup but before further interaction with the challenger) is necessary and allowed. This is important to simulate some possible attacks, such as the colluded internal attacks. Now we show the strength of Game 2 by tweaking a normal public key encryption that is secure in the standard formulation (where the challenger chooses the challenge identity or public key) in the following way. Suppose the challenger generates the private keys for parties following a particular rule, specifically, if the identities of parties satisfy a relation I, then their private keys are of a relation R. Suppose relation I is publicly known (so to the adversary) and polynomial-time checkable; moreover, given a subset of private keys of relation R, the other keys in the same set are polynomial-time computable. An example of relation R is a simple secret sharing scheme that the sum of n private keys in the set equals to a fixed value g. n-1 private keys are randomly sampled from the private key space. The n-th key is computed from the sum equation. The tweaked scheme is still secure if in the game, it is the challenger who chooses the challenge identity randomly, because the set of identities is substantially larger than the statically corrupted one. Hence it is negligibly likely that the corruption in advance will help the adversary recover the challenging party's private key randomly chosen by the challenger. However, we don't think that the tweaked scheme is secure in some sense. While allowing the adversary to choose the challenge identity, the tweaked scheme cannot be proved secure. This easily follows from the fact that the adversary can recover the challenging party's private key without explicitly querying the challenger and win the game with probability 1. The tweaked scheme also shows that there is a gap between the public key encryptions with and without allowing adaptive corruption. For example, if this time the challenger chooses the challenge identity ID_{ch}, the adversary can adaptively query other parties's private key in the same set (the queried identities and the challenge identity satisfy the relation I) to recover the private key of ID_{ch}, so to win the game.

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 is not 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. Specifically, if the challenger treats the adversary as a black-box, i.e., it only interacts with the adversary via an interface by answering just the queries defined in the game, the challenger of Game 1 cannot answer the decryption query without the corresponding private key. Otherwise, we can construct an adversary to use the challenger of Game 1 as a subroutine to win Game 2 defined in [3] (this is more obvious in the aforementioned single game model). Note that in the random oracle model, the challenger in the games indeed has extra advantages over the adversary if a random oracle is used, e.g., it can program the random oracle as its wish and has the full access to the complete input/output list (this ability is crucial to the plaintext-awareness defined in the random oracle) maintained by the random oracle (so we do not regard the reduction in the random oracle model as a black-box reduction). Hence it is possible that the challenger can answer such peculiar type of decryption query while the adversary cannot win Game 2 in the random oracle model. Note that the CL-PKE construction in [3] uses the random oracle in a substantial way to achieve plaintext awareness.

In this paper, we also use the random oracle and the plaintext awareness formulation to prove the security of our schemes, but we think a formulation that cannot be instantiated in the standard model has no great interest (note that there is a substantial difference between using random oracle to prove the security that can be achieved in the standard model and those cannot [5]). So, 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 which behaves honestly to follow the Decrypt algorithm, can decrypt the message successfully when it is used as a decryption oracle. Otherwise, the public key N'_B must be used in trivial means in the algorithm Encrypt. On the other hand, if a public key is used trivially in Encrypt, 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 the ciphertext somehow. In this case, the challenger will answer the decryption query correctly. In fact, this simulation maybe is still unnecessarily strong.

Although the security notion defined here seems weaker than the one in [3], it is guaranteed that the formulation can be instantiated in the standard model (without resorting to random oracle). An efficient IBE secure in the full model [8] was constructed by Waters [34] and a few efficient normal public-key encryptions achieving IND-CCA2 security were constructed in the literature, such as [16]. Yum and Lee presented a generic construction of certificateless encryption [36] by combining these two types of encryption (note that they also used a weaker formulation which does not require the challenger in Game 1 to answer a decryption query without the knowledge of the corresponding private key). Although Yum and Lee's reduction allows the adaptive corruption to Type-II adversaries, the majority of their argument still follows if this corruption query is removed. Note that the constructibility of the formulation does not imply that the constructions in this paper proved using the random oracle can definitely be instantiated.

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 [29], 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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, which has the following properties [8]:*

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 Weil and the modified Tate pairings on elliptic curves can be used to build such bilinear maps [33].

Assumption 1 Bilinear Diffie-Hellman Assumption (BDH) *[8] Let $\mathcal{G}$ be a parameter generator which with system parameters 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:*

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

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 Computation Diffie-Hellman Assumption (CDH) *Let $\mathcal{G}$ be a parameter generator which with system parameters 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:*

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

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

While in the commonly used setting for pairing-based schemes where $\mathbb{G}_1$ is the torsion group on an elliptic curve, the decision Diffie-Hellman problem (**DDH**) is simple. Specifically, given (P, aP, bP, cP), there is a polynomial-time algorithm to decide if $cP = abP$ by checking if the equation $\hat{e}(aP, bP) = \hat{e}(P, cP)$ holds. Hence essentially, we are using the Gap Diffie-Hellman assumption (**GDH**) that the CDH still appears hard, even in the presence of the DDH oracle [28].

2.3 Existing Schemes

Apart from introducing the notion of CL-PKC, Al-Riyami and Paterson also proposed a concrete scheme which extends the 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 $a, b, c \xleftarrow{R} \mathbb{Z}_q^*$, 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 were constructed as well in [3].

As the independent work, the same authors of [3] constructed a simple CL-PKE [4] presented on PKC 2005[3], which our CL-PKE scheme is very similar to. Despite the similarity, our work is different from [4] in a few ways. First, our formulation of CL-PKE is different from [4] and can be instantiated in the standard model. Second, our construction uses a hash function as a hinge to link two types of encryption scheme with a similar structure and is slightly simpler than the one in [4] (our scheme uses four hash functions instead of five in [4] and a tweaked version is even faster in some cases) and the reduction can be tighter as well. Third, we analyze the deficiency of the formulation of authenticated encryption and present a concrete construction.

It is not difficult to demonstrate that the scheme proposed by Gentry in [20] 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 $t_A P \in \mathbb{G}_1^*$

[3] We became aware of this work from the personal communication with one of the authors when their work was to be published, after we had posted the paper on IACR ePrint.

($t_A \in \mathbb{Z}_q^*$ is the private key $PrvKeyR$) and some other information (naturally, the identity ID_A). Second, algorithm Extract takes $info_A$, the **master-key** 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.

In [35], Yum and Lee presented a general construction of CL-PKE by an IBE and a traditional PKE following a formulation similar to [3], but which does not require the challenger in Game 1 to answer a decryption query without the knowledge of the corresponding private key. While, their model still allows the adaptive corruption in Game 2.

An authenticated CL-PKE was intended in [25] 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 $\langle x_1P, x_2P \rangle$ as $\langle X_B, Y_B \rangle$, the adversary can recover the decryption immediately by computing $T = x_1 x_A P$ and $\hat{e}(d_A, Y_B)^r = \hat{e}(rQ_A, x_2 P_{pub})$ in the proposed scheme.

3 The New CL-PKE

Pairing is a very heavy operation compared with the point scalar, exponentiation and hash operations. In the above two concrete 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 is exactly the integration, using a hash function as a hinge, of two algorithms with one of each type, i.e., the Boneh-Franklin's IBE [8] and a variant of ElGamal's Diffie-Hellman encryption scheme [18][1] strengthened using the Fujisaki-Okamoto's transform [19]. Using a hash function as a hinge to link two encryptions with similar structure can be generalized to some extent, such as the constructions in this paper.
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 **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.

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 proof of the scheme's property against Type-I adversaries mainly mimics the security proof of the Boneh-Franklin's IBE [8]. Note that as pointed out that by Galindo [21], the reduction of Lemma 4.6 in Boneh-Franklin's IBE [8] is not valid (the spotted error is more obvious if we mimic such a proof of our scheme because rP is used in H_2), so are Al-Riyami and Paterson's work [3][4] (their reductions can be fixed easily). While our proof has not such problem. The property against Type-II adversaries fairly straightly follows from the work in [19] based on the CDH assumption in $\mathbb{G}_1$ implied by the BDH assumption. The use of rP in H_2 is suggested in [31][16] which makes use of the Gap Diffie-Hellman assumption in $\mathbb{G}_1$ in the security proof to achieve non-malleability[4] [6] and a tighter reduction.

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.*

The proof is presented in Appendix A.

Note that the above scheme needs one more point scalar in both encryption and decryption algorithm compared with the Boneh-Franklin's IBE and the point scalar could be slower than the corresponding exponentiation of a root of the unity[5] of the extension field in some cases. For example, for a random $0 < r < q$ of 160 bits, by choosing the supersingular curve of embedding degree $k = 2$ defined

[4] Although proved in [6] 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 [31].

[5] The Weil pairing and the modified Tate pairing map two points to a root of unity of the corresponding extension field [33][10].

over the field $\mathbb{F}_p$ with 512-bit prime p to achieve the security level close to a 1024-bit RSA-based scheme, the point scalar needs about 1200 multiplications in $\mathbb{F}_p$ (according to Table IV.3. in [11] by assuming $I = 10M$ and the windowed NAF algorithm is used). While, the exponentiation of a p-th root of unity represented by a normal basis (which is of a special form [32]) can be computed with roughly 490 multiplications in $\mathbb{F}_p$ using the Lucas ladder (our experiment shows that the speed ratio of exponentiation to point scalar is about 4.5. Part of the reason of this ratio is because more modulo operations are used in point scalar, but p used in the experiment is not a Solinas prime and so, the cost of modulo is no longer trivial. Using the window method such as [26], the performance could be improved further). By sacrificing the bandwidth (for a longer ciphertext), we can tweak the above scheme to achieve better performance. In the tweaked scheme, an entity A publishes $N_A = \zeta^{t_A}$ where $\zeta = \hat{e}(P, P)$. The ciphertext is computed as $C = \langle rP, \zeta^r, \sigma \oplus H_2(rP, \zeta^r, g^r, f), m \oplus H_4(\sigma)\rangle$ where $f = N_A^r$ and g^r is computed as usual and the decryption algorithm is straightforward. As the extra component ζ^r can be computed from rP and P, the security proof follows the original scheme and by introducing one extra pairing $\hat{e}(rP, P)$ in the decryption algorithm, the ciphertext length can be reduced to the original size. Note that even with two extra exponentiations the tweaked scheme is still faster than the original one with the parameter example.

4 A CL-Auth-PKE

The new CL-PKE scheme can be simply extended to an authenticated encryption (Auth-PKE)(not signcryption) [24][2][37]. The formulation of identity-based Auth-PKE (with message privacy and authenticity property) can be found in [24], which can be extended to the certificatless setting with considering the extra public key. Due to limited space, we skip the definition of message authenticity, instead focus on a deficiency of message privacy definition in the literature.

The formulation and construction in [24] (and the *outsider security* in [2]) does not achieve the forward secrecy of sender's private keys (*forward secrecy* for short). In particular, if the sender's private keys are compromised, any message encrypted with these keys could be recovered. Moreover, because lacking of sender-key forward secrecy implies that the sender can decrypt and accept a message encrypted by itself (so the message authenticity is breached in some sense), a scheme can be attacked. For example, if A sends B a message "Please transfer \$1000 to C's account" protected by Lynn's Auth-PKE [24], C can eavesdrop this message and replays it to A. A will be to able to decrypt this message and accept it as a valid message from B. So, C will get \$2000 with \$1000 bonus. (Note that this attack abuses the symmetry of the used pairing.) 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 has compromised 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 victim entity cannot notice the fact that its keys were leaked). The adversary wants to decrypt the messages auth-encrypted after the compromise of the 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[6] 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 [19] is inadequate).

To fill in the gap we define the forward secrecy of authenticated encryption here. 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.
  - 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_s 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: $Adv_{\mathcal{E},\mathcal{A}}^{I-CPA}(k) = |Pr[b' = b] - 1/2|$. Note that, the above game does not allow the adversary to adaptively choose the sender or receiver (in the *forward secrecy* setting). Similarly, we can design 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

[6] In this case, we need the insider message privacy but outsider message authenticity formulated in [2].

only if it achieves message privacy, message authenticity 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 $m \in \mathcal{M}$, the identity ID_A, the public key N_A, the private keys d_A, t_A of sender A, the identity ID_B and the 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 $f = rt_AN_B$.
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 $g' = \hat{e}(U, d_B)$, $f' = t_BT$ 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 [8], 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 the Fujisaki-Okamoto's transform is IND-CCA secure, we can prove that the scheme is forward secure against Type-II adversaries. The message authenticity property can be proved using a similar reduction as in [24]. Intuitively, without the sender and receiver's private key, given a plaintext, the adversary cannot generate a valid ciphertext (the Type-I adversary needs to find $\langle rQ_A, \hat{e}(Q_A, Q_B)^{rs}\rangle$ given $\langle P, sP, Q_A, Q_B\rangle$, while the Type-II adversary needs to find $\langle rN_A, rt_AN_B\rangle$ given $\langle P, N_A, N_B\rangle$). Moreover the scheme is adaptively secure (IND-CCA2) which implies non-malleability, this is, given a ciphertext without knowing the plaintext, the adversary cannot generate another valid ciphertext whose corresponding plaintext has a relation with the plaintext of the first ciphertext. Hence the *outsider* unforgeablility (message authenticity) is achieved.

We 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, which cannot be achieved by Lynn's scheme. In this case, the sender's identity is part of the message being auth-encrypted. This is very useful in the environments that the sender wants to hide its identity.

5 Remarks on CL-PKE's

First, we evaluate the complexity of the CL-PKE schemes. The schemes have four major operations, i.e., Pairing, Scalar, Exponentiation and Hash. Without pre-computation, 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 [10]. 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 the first three schemes have the same ciphertext length and both the tweaked CL-PKE and the Auth-Encryption scheme have one more component.

	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**
Tweaked New scheme	1	1	3	4	1	1	2^*	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 [23][3][20]), 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 an entity 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. But the certificate is going to be used as the entity's another private 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 [31][16] (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 [20] 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 can be traced (one advantage of Gentry's $PrvKeyL$ generation method). This may not be 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 a similar fashion as Gentry's, e.g., $PrvKeyL_A = sH_1(ID_A \| N_A)$ (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 the key evolvement scheme [20], intrusion-resilient encryption [17], mediated encryption [7], etc, have been attempted. None of them can fully solve the problem.

6 Acknowledgement

The first author would like to thank Kenneth G. Paterson for his comments on the draft of this paper.

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 CT-RSA 2001, LNCS 2020, 2001.
2. J. H. An, Y. Dodis and T. Rabin. On the security of joint signature and encryption. in Advances in Cryptology-Eurocrypt 2002, LNCS 2332, 2002.
3. S. S. Al-Riyami and K. G. Paterson. Certificateless Public Key Cryptography. in Advances in Cryptology-Asiacrypt 2003, LNCS 2894, 2003. See also Cryptology ePrint Archive, Report 2003/126.

4. S. S. Al-Riyami and K. G. Paterson. CBE from CL-PKE: A Generic Construction and Efficient Schemes. to appear in PKC 2005.
5. M. Bellare, A. Boldyreva and A. Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. in Advances in Cryptology-EUROCRYPT 2004, LNCS 3027, 2004.
6. 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 1462, 1998.
7. D. Boneh, X. Ding, G. Tsudik and C. Wong. A Method for Fast Revocation of Public Key Certificates and Security Capabilities. in Proceedings of the 10th USENIX Security Symposium, USENIX, 2001.
8. D. Boneh and M. Franklin. Identity Based Encryption from The Weil Pairing. extended abstract in Advances in Cryptology-Crypto 2001, LNCS 2139, 2001.
9. D. Beaver and S. Haber. Cryptographic Protocols Provably Secure Against Dynamic Adversaries. in Advance in Cryptology-Eurocrypt 1992, LNCS 658, pp. 307-323, 1992.
10. P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. in Advances in Cryptology-Crypto 2002, LNCS 2442, pp. 354-368, 2002. See also Cryptology ePrint Archive, Report 2002/008.
11. I.F. Blake, G. Seroussi and N. P. Smart. Elliptic Curve in Cryptography", Cambridage Press, 1999.
12. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. 42nd IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, pp. 136-145, 2001.
13. R. Canetti, U. Feige, O. Goldreich and M. Naor. Adaptively Secure Computation. in 28th ACM Symposium on Theory of Computing (STOC), ACM, pp. 639-648, 1996. Full version in MIT-LCS-TR #682, 1996.
14. R. Canetti, S. Halevi and J. Katz. A Forward-Secure Public-Key Encryption Scheme. in Advances in Cryptology-Eurocrypt 2003, LNCS 2656, 2003
15. R. Canetti, S. Halevi and J. Katz. Adaptively-Secure, Non-Interactive Public-Key Encryption. To Appear in TCC 2005. Cryptology ePrint Archive, Report 2004/317.
16. 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.
17. Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung. Intrusion-Resilient Public-Key Encryption. in CT-RSA 2003, LNCS 2612, pp. 19-32, 2003.
18. T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31:469-472, 1985.
19. E. Fujisaki and T. Okamotom. Secure Integration of Asymmetric and Symmetric Encryption Schemes. in Advances in Cryptology-CRYPTO 1999, LNCS 1666, pp.535-554, 1999.
20. C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. Cryptology ePrint Archive, 2003/183.
21. D. Galindo. Improved Identity Based Encryption. 2005. Available on http://www.cs.ru.nl/ dgalindo/improvedIBEfull.pdf.
22. S. Goldwasser and S. Micali. Probabilistic encryption. Journal of Computer and System Sciences Vol. 28, 270-299, 1984.
23. C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. in Proceedings of Asiacrypt 2002, LNCS 2501, 2002.
24. B. Lynn. Authenticated Identity-Based Encryption. Cryptology ePrint Archive, Report 2002/072.

25. Y.-R. Lee and H.-S. Lee. An Authenticated Certificateless Public Key Encryption Scheme. Cryptology ePrint Archive, 2004/150.
26. B. Möller. Improved Techniques for Fast Exponentiation. in Information Security and Cryptology-ICISC 2002, LNCS 2587, 2003.
27. J.B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case. in Advance in Cryptology-Crypto 2002, LNCS 2442, pp. 111-126, 2002.
28. T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. in Practice and Theory in Public Key Cryptography-PKC'2001, LNCS 1992, 2001.
29. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. in Advances in Cryptology-Crypto 1984, LNCS 196, 1984.
30. V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. in Advances in Cryptology-EUROCRYPT 2000, LNCS 1807, 2000.
31. V. Shoup. A Proposal for an ISO Standard for Public Key Encryption. 2001.
32. M. Scott and P. S. L. M. Barreto. Compressed Pairings. in Advances in Cryptology-Crypto 2004, LNCS 3152, 2004. See also Cryptology ePrint Archive, Report 2004/032.
33. E. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. in Advances in Cryptology-Eurocrypt 2001, LNCS 2045, pp. 195-210, 2001.
34. B. R. Waters. Efficient Identity-Based Encryption Without Random Oracles. Cryptology ePrint Archive, Report 2004/180.
35. D.H. Yum and P.J. Lee. Identitiy-based cryptography in public key management. in EuroPKI 2004, LNCS 3093, pp. 71-84, 2004.
36. D.H. Yum and P.J. Lee. Generic construction of certificateless encryption. in ICCSA 2004, LNCS 3043, pp. 802-811, 2004.
37. Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). in Advances in Cryptology-Crypto 1997, LNCS 1294, 1997.

Appendix A

Proof of Theorem 1.

The proof of the security of the new CL-PKE follows from Proposition 1 and 2.

The proof of Proposition 1 use a similar strategy of the Boneh-Franklin's proof [8]. Similarly we first define two basic schemes **BasicPub** and $\textbf{BasicPub}^{\textbf{hy}}$.

Definition 3 BasicPub
BasicPub *is specified by three algorithms:* **keygen**, **encrypt** *and* **decrypt**.
keygen: *Given a security parameter k, the parameter generator follows the steps.*

1. *Generate two cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ 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 $P_{pub} = sP$. Choose a random $Q_A \in \mathbb{G}_1^*$.*
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 $n > 0$.*

The message space is $\mathcal{M} = \{0,1\}^n$*. The ciphertext space is* $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n$*. The public* **params** *is* $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_A, H_2\rangle$ *and the private key is* $d_A = sQ_A$*.*
encrypt: *Given a plaintext* $m \in \mathcal{M}$*, an element* $N \in \mathbb{G}_1^*$ *and the public* **params**, *choose a random* $r \in \mathbb{Z}_q^*$ *and compute ciphertext* $C = \langle rP, m \oplus H_2(rP, g^r, rN)\rangle$*, where* $g = \hat{e}(P_{pub}, Q_A)$*.*
decrypt: *Given a ciphertext* $C = \langle U, V\rangle$*, an element* $t \in \mathbb{Z}_q^*$*, the public* **params**, *and the private key* d_A*, compute* $g' = \hat{e}(U, d_A)$ *and plaintext* $m = V \oplus H_2(U, g', tU)$*.*

Definition 4 BasicPub$^{\mathbf{hy}}$
BasicPub$^{\mathbf{hy}}$ *is specified by three algorithms:* **keygen**, **encrypt**, **decrypt**.
keygen: *This algorithm is identical to* **keygen** *of* **BasicPub**, *except that it chooses two additional hash functions* $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ *and* $H_4 : \{0,1\}^n \to \{0,1\}^n$*.*
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 public* **params** *is* $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_A, H_2, H_3, H_4\rangle$ *and the private key is* $d_A = sQ_A$*.*
encrypt: *Given a plaintext* $m \in \mathcal{M}$*, an element* $N \in \mathbb{G}_1^*$ *and the public* **params**, *follow the steps:*

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

decrypt: *Given a ciphertext* $C = \langle U, V, W\rangle$*, an element* $t \in \mathbb{Z}_p^*$*, the public* **params**, *and the private key* d_A*, follow the steps.*

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

Proposition 1 *The CL-PKE scheme is secure against the Type-I adversaries provided that* H_1, H_2, H_3*, and* H_4 *are random oracles and the BDH assumption is sound.*

Proof: The proof follows directly from the following Lemma 1, 2, 3.

Lemma 1 *Suppose that* H_1 *is a random oracle and that there exists a Type-I IND-CCA2 adversary* $\mathcal{A}$ *against CL-PKE with advantage* $\epsilon(k)$ *which makes at most* q_1 *queries to* H_1*. Then there exits a Type-I IND-CCA2 adversary* $\mathcal{B}$ *against* **BasicPub$^{\mathbf{hy}}$** *with advantage at least* $\epsilon(k)/q_1$ *which runs in time O(time(*$\mathcal{A}$ *)).*

Proof: We construct an IND-CCA2 adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage against **BasicPub$^{\mathbf{hy}}$**. The game between challenger $\mathcal{C}$ and adversary $\mathcal{B}$ starts with the challenger first generating a random public **params** by running algorithm **keygen** of **BasicPub$^{\mathbf{hy}}$**. The result is a public **params** $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_A, H_2, H_3, H_4\rangle$ and a private key $d_A = sQ_A$ where $s \in \mathbb{Z}_q^*$. The challenger passes K_{pub} to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ mounts an IND-CCA2 attack on the **BasicPub$^{\mathbf{hy}}$** with **params** K_{pub} using the help of $\mathcal{A}$ as follows.

$\mathcal{B}$ chooses an index I with $1 \le I \le q_1$ and simulates the algorithm **Setup** of CL-PKE for $\mathcal{A}$ by supplying $\mathcal{A}$ with the CL-PKE's public **params**= $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$ where H_1 is a random oracle controlled by $\mathcal{B}$. $\mathcal{B}$ uses value s as the **master-key** which it does not know. Adversary $\mathcal{A}$ can make queries of H_1 at any time. These queries are handled by the algorithm **H_1-queries**.

H_1-queries (ID_i)**:** $\mathcal{B}$ maintains a list of tuples $\langle ID_j, Q_j, h_j \rangle$ as explained below. We refer to this list as H_1^{list}. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point ID_i, $\mathcal{B}$ responds as follows:

1. If ID_i already appears on the H_1^{list} in a tuple $\langle ID_i, Q_i, h_i \rangle$, then $\mathcal{B}$ responds with $H_1(ID_i) = Q_i \in \mathbb{G}_1^*$.
2. Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $\langle ID_I, Q_A, \perp \rangle$ into the tuple list and responds with $H_1(ID_I) = Q_A$.
3. Otherwise, $\mathcal{B}$ selects a random integer h_i from $\mathbb{Z}_q^*$, computes $Q_i = h_i P$, and stores $\langle ID_i, Q_i, h_i \rangle$ into the tuple list. $\mathcal{B}$ responds with $H_1(ID_i) = Q_i$.

Phase 1: $\mathcal{A}$ launches Phase 1 of its attack, by make a series of requests, each of which is either a $PrKeyL$ extraction, a publish query, a replace query, a $PrKeyR$ extraction or a decryption query. $\mathcal{B}$ replies to these requests as follows (we assume that an H_1 query on the related identity has been issued.):

Extract $PrKeyL$ **on** (ID_i)**:** If $ID_i = ID_I$, $\mathcal{B}$ aborts the game (**Event 1**); otherwise $\mathcal{B}$ responds with $h_i sP$ where h_i is from the tuple corresponding to ID_i in H_1^{list}.

Publish query on (ID_i)**:** To respond to this query, $\mathcal{B}$ maintains another list K_{list} with tuples of the form $\langle ID_i, t_i, t_i P, R_i \rangle$ (ID_i is the index). For a new ID_i, $\mathcal{B}$ randomly chooses $t_i \in \mathbb{Z}_q^*$ and inserts the tuple $\langle ID_i, t_i, t_i P, t_i P \rangle$ into the list; otherwise, $\mathcal{B}$ responds with R_i.

Replace query on (ID_i) **with** N_i**:** $\mathcal{B}$ replaces R_i of the tuple indexed by ID_i on K_{list} with N_i.

Extract $PrKeyR$ **on** (ID_i)**:** $\mathcal{B}$ checks if $R_i = t_i P$ of the tuple indexed by ID_i on the list K_{list}. If $R_i = t_i P$, $\mathcal{B}$ returns t_i; otherwise $\mathcal{B}$ aborts the game (**Event 2**).

Decryption Query (ID_i, C_i, N_i)**:** $\mathcal{B}$ first searches K_{list} to find a tuple with $N_i = t_i P$. If such tuple does not exist, $\mathcal{B}$ aborts the game (**Event 3**). If the request is to decrypt $C_i = \langle U, V, W \rangle$ on ID_I (i.e., $ID_i = ID_I$), then $\mathcal{B}$ sends the decryption query $\langle U, V, W \rangle$ and t_i to $\mathcal{C}$ and simply relays the plaintext got from $\mathcal{C}$ to $\mathcal{A}$ directly. Otherwise $\mathcal{B}$ tries to perform the decryption by first computing $g^r = \hat{e}(U, h_i sP)$ and $t_i U$, then querying $H_i = H_2(U, g^r, t_i U)$ (H_2 is controlled by $\mathcal{C}$).

Challenge Phase: At some point, $\mathcal{A}$ decides to end Phase 1 and picks ID_{ch}, N_{ch} and two messages m_0, m_1 on which it wants to be challenged. If $ID_{ch} \neq ID_I$ then $\mathcal{B}$ aborts the game (**Event 4**). Otherwise $\mathcal{B}$ passes $\mathcal{C}$ the pair of m_0, m_1 as the messages on which it wishes to be challenged and N_{ch} as well. $\mathcal{C}$ responds with the challenger ciphertext $C_{ch} = \langle U', V', W' \rangle$. Then $\mathcal{B}$ forwards C_{ch} to $\mathcal{A}$.

Phase 2: $\mathcal{B}$ continues to respond to requests in the same way as it did in Phase 1. But if any decryption query is issued on $\langle ID_I, C_{ch}, N_{ch}\rangle$, then $\mathcal{B}$ aborts (**Event 5**).
Guess: $\mathcal{A}$ makes a guess b' for b. $\mathcal{B}$ outputs b' as its own guess.

Claim: If the algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}$'s view is identical to its view in the real attack.

Proof: $\mathcal{B}$'s responses to H_1 queries are uniformly and independently distributed in $\mathbb{G}_1^*$ as in the real attack. All responses to $\mathcal{A}$'s requests are valid, if $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext $\langle U', V', W'\rangle$ is a valid encryption in CL-PKE for m_b where $b \in \{0,1\}$ is random.

The remaining problem is to calculate the probability that $\mathcal{B}$ does not abort during simulation. The algorithm $\mathcal{B}$ could abort when one of the following events happens: (1) **Event 1**, denoted as $\mathcal{H}_1$: $\mathcal{A}$ queried $PrKeyL$ for ID_I at some point; (2) **Event 2**, denoted as $\mathcal{H}_2$: $\mathcal{A}$ replaced a public key of an entity, but queried its $PrKeyR$ (this is not allowed in the game). (3) **Event 3**, denoted as $\mathcal{H}_3$: $\mathcal{A}$ asked a decryption query under an unknown public key N_i (this is not allowed in the game). (4) **Event 4**, denoted as $\mathcal{H}_4$: $\mathcal{A}$ did not choose ID_I as ID_{ch}; or (5) **Event 5**, denoted as $\mathcal{H}_5$: $\mathcal{B}$ relayed a decryption query on C_{ch} to $\mathcal{C}$ in phase 2.

Because of the way that $\mathcal{B}$ forwards ciphertexts, the last event implies that $\mathcal{A}$ didn't choose ID_I. We also notice that the event $\neg\mathcal{H}_4$ implies that the event $\neg\mathcal{H}_1$. Hence we have

$$Pr[\mathcal{B}\text{ does not abort}] = Pr[\neg\mathcal{H}_1 \wedge \neg\mathcal{H}_4 \wedge \neg\mathcal{H}_5] = Pr[\neg\mathcal{H}_4] \geq 1/q_1.$$

So, the lemma follows.

Lemma 2 *Let H_3, H_4 be random oracles. Let $\mathcal{A}$ be a Type-I IND-CCA2 adversary against* **BasicPub$^{\mathbf{hy}}$**. *Suppose $\mathcal{A}$ has running time $t(k)$, makes at most q_D decryption queries, and makes q_3 and q_4 queries to H_3 and H_4 respectively. If $\mathcal{A}$ has non-negligible advantage $\epsilon(k)$ against* **BasicPub$^{\mathbf{hy}}$**, *then there exists a Type-I IND-CPA (chosen plaintext attack [6]) adversary $\mathcal{B}$ against* **BasicPub** *with with advantage $\epsilon_1(k)$ and running time $t_1(k)$ where*

$$\epsilon_1(k) \geq \frac{1}{2(q_3+q_4)}[(\epsilon(k)+1)(1-\tfrac{2}{q})^{q_D} - 1]$$
$$t_1(k) \leq t(k) + O((q_3+q_4)\cdot(n+\log q)).$$

Proof: This lemma follows from the result of the Fujisaki-Okamoto's transform [19]. The extra N in the scheme does not affect the reduction, because H_2 is a random oracle.

Lemma 3 *Let H_2 be a random oracle. Suppose there exists a Type-I IND-CPA adversary $\mathcal{A}$ against* **BasicPub** *which has non-negligible advantage $\epsilon(k)$ and queries H_2 at most q_2 times. Then there exists an algorithm $\mathcal{B}$ to solve the BDH problem with non-negligible advantage $2\epsilon(k)/q_2$.*

Proof: The proof is similar to Lemma 4.3 in [8]. Algorithm $\mathcal{B}$ is given as input the BDH parameters $\langle q, \mathbb{G}_1, \mathbb{G}_2, P, \hat{e} \rangle$ produced by $\mathcal{G}$ and a random instance $\langle P, aP, bP, cP \rangle$ where a, b, c are random in $\mathbb{Z}_q^*$. Algorithm $\mathcal{B}$ finds $\hat{e}(P,P)^{abc}$ by interacting with $\mathcal{A}$ as follows:

Algorithm $\mathcal{B}$ simulates algorithm **keygen** of **BasicPub** to create the public **params** $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_A, H_2 \rangle$ by setting $P_{pub} = aP$ (i.e., $s = a$), $Q_A = bP$. H_2 is a random oracle controlled by $\mathcal{B}$. So, the private key is $d_A = abP$ which $\mathcal{B}$ does not know. Algorithm $\mathcal{B}$ passes the public **params** K_{pub} to $\mathcal{A}$ and responds queries as follows.

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

1. If on the list there is a tuple indexed by (X_i, Y_i, Z_i), then $\mathcal{B}$ responds with H_i.
2. Otherwise, $\mathcal{B}$ randomly chooses a string $H_i \in \{0,1\}^n$ and inserts a new tuple (X_i, Y_i, Z_i, H_i) to the list. It responds to $\mathcal{A}$ with H_i.

Challenge: Algorithm $\mathcal{A}$ outputs two messages m_0, m_1 and an element $N \in \mathbb{G}_1^*$ on which it wants to be challenged. $\mathcal{B}$ chooses a random string $R \in \{0,1\}^n$, and defines $C_{ch} = \langle U', V' \rangle = \langle cP, R \rangle$. $\mathcal{B}$ gives C_{ch} as the challenge to $\mathcal{A}$. Observe that the decryption of C_{ch} is

$$V' \oplus H_2(U', \hat{e}(U', d_A), cN) = R \oplus H_2(cP, \hat{e}(cP, abP), cN)$$

Guess: Algorithm $\mathcal{A}$ outputs its guess $c' \in \{0,1\}$. At this point $\mathcal{B}$ picks a random tuple $\langle X_i, Y_i, Z_i, H_i \rangle$ from the set which contains the tuples from H_2^{list} with the form $X_i = cP$. $\mathcal{B}$ outputs Y_i.

Let $\mathcal{H}$ be the event that algorithm $\mathcal{A}$ issues a query for $H_2(cP, \hat{e}(cP, abP), cN))$ at some point during the simulation above. Using the same methods in [8], we can prove the following two claims:

Claim 1: $\Pr[\mathcal{H}]$ in the simulation above is equal to $\Pr[\mathcal{H}]$ in the real attack.

Claim 2: In the real attack we have $Pr[\mathcal{H}] \geq 2\epsilon(k)$.

Assume that $\mathcal{A}$ has queried q_2 distinct value on H_2. Following from the above two claims, we have that $\mathcal{B}$ produces the correct answer with probability at least $2\epsilon(k)/q_2$.

If we separate the queries to H_2 into two stages, i.e., before the challenge phase (q_2^a times) and after the challenge phase (q_2^p times), then we can obtain a tighter reduction $2\epsilon(k)/q_2^p$, because cP is randomly chosen and it is negligibly likely that $\mathcal{A}$ has queried with this value before the challenge phase.

Proposition 2 *The CL-PKE scheme is secure against the Type-II adversaries provided that H_1, H_2, H_3, and H_4 are random oracles and the GDH assumption in $\mathbb{G}_1$ is sound.*

The proof of Proposition 2 is quite similar to the following reduction of CL-Auth-PKE's forward secrecy. We omit the details.

□

Proof of forward secrecy of the CL-Auth-PKE.

We can follow the method of Lemma 4.3 in [8] to prove that the CL-Auth-PKE achieves the forward secrecy against Type-I adversaries.

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 random 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 $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, n, \hat{e}, P, P_{pub}, Q_A, t_A P, Q_B, t_B P, H_2 \rangle$. 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 $f = rt_A t_B P$.
2. Set the ciphertext to $C = \langle rt_A P, rQ_A, m \oplus H_2(rt_A P, rQ_A, g^r, f)\rangle$.

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

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

Lemma 4 *Let H_2 be a random oracle. Let $\mathcal{A}$ be a sender-key-known IND-CPA adversary that has advantage $\epsilon(k)$ against* **BasicAuthPub**. *Suppose $\mathcal{A}$ makes a total of q_{H_2} 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 ease to prove the lemma, we extend the BDH assumption to the following assumption (EBDH). Given $\langle q, \mathbb{G}_1, \mathbb{G}_2, R, aR, bR, cR, vR, vaR, vcR, t_1, t_1 vR, t_1 vcR, t_2, t_2 vR\rangle$ 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 $\hat{e}(R,R)^{abc}$. One can easily verify that the BDH and the EBDH assumption 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 $\langle q, \mathbb{G}_1, \mathbb{G}_2, n, \hat{e}, vR, vaR, R, t_1 vR, bR, t_2 vR, H_2\rangle$, 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 respond to 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 respond to 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 [8], 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, $\mathcal{B}$ can 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.

$\square$

Lemma 4 shows that the **BasicAuthPub** scheme already achieves the forward secrecy again Type-I adversaries. By applying the 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 [8].

Appendix B

An ECIES-KEM-similar hybrid encryption [1][31] 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 **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 publish 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 $m = ENC_{k_1}^{-1}(V)$ as the plaintext.