Token-Controlled Public Key Encryption

Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo

Centre for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong,
Wollongong NSW 2522, Australia
{baek, rei, wsusilo}@uow.edu.au

Abstract. "Token-controlled public key encryption" is a public key encryption scheme where individual message can be encrypted and sent to every receiver, but the receiver cannot decrypt the message until he/she is given an extra piece of information called a "*token*". The token will not reveal any information about the messages that the sender originally sent and the communication overhead for releasing the token is very small. Also, it is possible that a single token can control the decryption of a number of ciphertexts sent to multiple receivers. We formalize security model for such scheme and show efficient and provably secure constructions based on known computational assumptions in the random oracle model.

1 Introduction

Consider the following scenario. A company ABC wants to send a payslip to its employees, in such a way that each employee can decrypt the authorization code to obtain his/her cash *at the appointed time*. The description of the payslip (such as the amount, the detail of the employee, etc.) can be read when the payslip is received, but the authorization code that will allow the bank to transfer the money to the employee's account will only be available at the appointed time, i.e. on the pay-day. Intuitively, this problem can be solved easily by sending the payslip and authorization code on the pay-day to each employee. However, due to the large number of employees in the company, it would be more convenient if the information can be sent progressively, i.e. throughout the week before the pay day occurs. Nevertheless, the company do not want to allow their employees to obtain their salary before the pay day.

Formally, we would like to obtain a new public key encryption scheme, which we call a "*token-controlled public key encryption scheme*", where individual message can be encrypted and sent to every receiver, but the receiver cannot decrypt the message until he/she is given an extra piece of information, which we call a "*token*", from the third party that was appointed by the sender. Additionally, the token will not reveal any information about the messages that the sender originally sent. We also require that the communication and computational overhead for releasing the token to be small. For example, after its release by the company, *one* token whose size is similar to that of the security parameter within the system, e.g. 1024 bits, can control the decryption of the bank authorization codes different from employee to employee.

We argue that such a scheme has many other real world applications. Suppose there is a millionaire who would like to write a will for his three sons. This will is written, sealed and sent to his sons. However, he will not allow his sons to read the will before he passes away, and therefore, he will send the token used to create the sealed will to a lawyer that he trusts. On one hand, this lawyer cannot read the will because he has no access to this encrypted document, and on the other hand, the three sons cannot read the will since they require the key information that is held by the lawyer.

2 Related Work and Our Contributions

Related Work. The closest work related to our research is the "timed-release cryptography" a.k.a. "sending information into the future" discussed in the papers of May [8], Rivest et al. [10], and Di Crescenzo et al. [6]. The approaches made in those papers to realize the timed-cryptography can be categorized into a "computational approach" and an "agent-based approach". The idea of the computational approach of [10] is that the time to recover a secret is given by the minimum computational effort required by any machine to perform some computation which will enable one to recover the secret. In [10], a computational primitive based on the hardness of the integer factorization was proposed to realize the idea, subsequently, a timed-release encryption scheme was constructed. However, as discussed in the same paper, this primitive does not give precise time-release as computational capability can vary from machine to machine.

On the other hand, in the agent-based approach, a trusted third party is expected to release a secret at the appointed time. As a result, precise time-release is possible. The schemes in [8] and [6], and the second scheme in [10] follow this approach. In [8], it was suggested that a cryptographic key K should be stored by a trusted agent while the encryption of the message M with K, denoted by $C = E(K, M)$, is sent to the receiver. At time t, the agent releases K which will enable the receiver to decrypt C. However, one of the drawbacks in this approach is that the agent must store the keys for all users. Rivest et al.'s [10] scheme resolves this problem by having the agent use its own key S to encrypt the key K requested by the sender and yield $C' = E(S, K)$. $C = E(K, M)$ and C' are then sent to the receiver. At the time when the agent releases S, the receiver recovers K from C' and subsequently decrypt C. In our point of view, the main drawback of the approaches of [8] and [10] is that in order to protect confidentiality of the message M, the encryption key K or the agent's secret key S should be delivered to the receiver through secure channels after the release-time has passed. If K and S are just published as described in [8] and [10], no secrecy on the message M is guaranteed.

In contrast to the above methods, the approach of [6] does not have this problem. In the timed-release encryption scheme proposed in [6], a plaintext message M

is encrypted by the receiver's public key pk_R and the resulting ciphertext is re-encrypted under the agent's public key pk_A. The resulting encryption, which we denote by $C = E(pk_A, E(pk_R, M))$, together with the release-time is sent to the receiver. On receiving C and the time information, the receiver interacts with the agent using the computationally intensive "conditional oblivious transfer protocol" which will give the receiver $E(pk_R, M)$ if the release-time is not less than the current time of the agent, otherwise, gives nothing to the receiver. Since the message is encrypted by the receiver's public key and this can only be decrypted by the receiver who has the corresponding private key after the release-time has passed, the problems in [8] and [10] do not happen. This property is in fact what we also need in our token-controlled public key encryption, but other requirements for it is not exactly the same as those for the scheme in [6] as outlined below.
Our Contributions. Based on the scenarios illustrated in Section 1 and the discussions on the related work, we summarize the features of our token-controlled public key encryption scheme:

1. A *token* independently chosen from a decryption key (a receiver's private key) is revealed by the third party so that the confidentiality of the plaintext message against parties except for the correct receiver is attained even after the token is released.
2. The receiver does not have to be involved in a computationally-heavy protocol to obtain a token.
3. A token can be *reusable* in the multiple receiver setting. That is, *one token* can control the decryption operation of multiple receivers without compromising security.

In the rest of the paper, we define a formal security model for the single receiver token-controlled public key encryption and then propose efficient and provably secure schemes in the random oracle model [3].
Remark. We remark that the security model and the scheme for the single user setting can readily be extended to the multiple receiver setting. In this setting, "token reusability" stated above is achieved. However, we omit them in the current version of the paper due to the lack of space. We also omit the proofs of the theorems presented in the current version, except for the proof of Theorem 1. The full version of this paper will contain all the omitted proofs.

3 Formal Security Model for the Single Receiver Setting

Generic Description. We begin with a high level description. In a token-controlled public key encryption scheme, the sender first picks a token at random from predefined token space. The sender then encrypts a plaintext message using the receiver's public key and the token, and sends the ciphertext to the receiver. Additionally, the sender gives the token to a "semi-trusted" third party. What we mean by "semi-trusted" here is that the third party is not required to store any private keys from the sender and the receiver, but is required to release a token honestly at the time previously requested by the sender. We can also assume that this party may behave maliciously to break the confidentiality of a ciphertext given to the receiver. (This will be discussed further in Section 3). *Only when the token is released (or published)* by the third party, the receiver can decrypt the ciphertext.

Compared with the model in [6], our token-controlled public key encryption does not provide "sender-anonymity" against a third party in that the sender must contact the third party to hand in a token. This is different from the model in [6] which resulted in a scheme that requires a computationally intensive protocol between the receiver and the third party. We now formally define a token-controlled public key encryption scheme as follows.

Definition 1 (TCPKE). A Token-Controlled Public Key Encryption scheme denoted by "TCPKE" consists of the following algorithms.

- A Randomized Key Generation Algorithm $\mathsf{GK}^{\mathsf{TCPKE}}(k)$: Taking a security parameter $k \in \mathbb{N}$ as input, this algorithm returns a private and public key pair (sk, pk). Note that pk includes the security parameter, descriptions of a finite plaintext space $\mathcal{P}$, a finite token space $\mathcal{T}$, and a ciphertext space $\mathcal{C}$.
- A Randomized Token Generation Algorithm $\mathsf{GT}^{\mathsf{TCPKE}}(k)$: Taking a security parameter $k \in \mathbb{N}$ as input, this algorithm chooses a token $tk \in \mathcal{T}$ at random and returns it.
- A Randomized Encryption Algorithm $\mathsf{E}^{\mathsf{TCPKE}}(pk, tk, M)$: Taking pk, tk, and $M \in \mathcal{P}$ as input, this algorithm returns a ciphertext $C \in \mathcal{C}$ which is an encryption of M.
- A Decryption Algorithm $\mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$: Taking sk, tk, and C as input, this algorithm returns a decryption D, which is either a plaintext $M \in \mathcal{P}$ or a special symbol "*Reject*".

Security against Outside Attackers: IND-TCPKE-T1CCA/T2CCA In terms of security of TCPKE, we need to consider two types of attackers, "*outside*" and "*inside*" attackers. This categorization is based on whether the attacker possesses the receiver's private key or not.

[*Type-1 Outside Attacker.*] We can further categorize the outside attackers into "type-1 outside attackers" and "type-2 outside attackers". Holding the public key of the receiver, the type-1 outside attacker's goal is to break the confidentiality of a TCPKE ciphertext created under a fixed token called a "target token" which is not given to the attacker. We assume that the attack conducted by this attacker is very active. First, the attacker has access to a "*token-embedded*" encryption oracle, which, upon receiving a plaintext message, returns a corresponding ciphertext created under the target token and the public key of the receiver. Also, this attacker has access to a decryption oracle, which, upon receiving a token-ciphertext pair of the attacker's choice, returns a corresponding decryption. Note that this models a situation where the attacker can record all the previous tokens and ciphertexts communicated among the sender, the receiver, and the third party to attack current ciphertexts.

Below, we formally define a security notion for TCPKE against the type-1 outside attacker, which we call "IND-TCPKE-T1CCA".

Definition 2 (IND-TCPKE-T1CCA). Let A be an attacker whose running time is bounded by t which is polynomial in a security parameter k. We now consider the following game:

> **Phase 1**: The key generation algorithm $\mathsf{GK}^{\mathsf{TCPKE}}(k)$ and the token generation algorithm $\mathsf{GT}^{\mathsf{TCPKE}}(k)$ are run. A private and public key pair (sk, pk) and a target token tk^* are then generated. pk is given to A while tk^* and sk are kept secret from A.
> **Phase 2**: A queries a number of plaintexts, each of which is denoted by M, to the token-embedded encryption oracle and obtains a corresponding ciphertext $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M)$. A also queries a number of token-ciphertext pairs, each of which is denoted by (tk, C), to the decryption oracle and obtains a corresponding decryption $D = \mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$, which is either a plaintext or a "*Reject*" symbol.
> **Phase 3**: A outputs two equal-length plaintexts (M_0, M_1). $\beta \in \{0, 1\}$ is then chosen at random and a target ciphertext $C^* = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M_\beta)$ is created and returned to A.
> **Phase 4**: A issues a number of encryption and decryption oracle queries as in Phase 2.
> **Phase 5**: A outputs its guess $\tilde{\beta} \in \{0, 1\}$.

We define A's success by the probability $\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k) = 2 \cdot \Pr[\tilde{\beta} = \beta] - 1$. The TCPKE scheme is said to be IND-TCPKE-T1CCA secure if $\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k)$ is negligible in k.

Note in the above attack game that no restriction is imposed on the token and ciphertext pairs queried to the decryption oracle in Phase 4. As a result, it is possible that $(tk, C) = (tk^*, C^*)$. However, if this event happens, the attacker comes to know β with probability 1 and hence the IND-TCPKE-T1CCA is broken. Nevertheless, this would show that the token must be chosen from an exponentially large space, which makes the probability that query tk equals tk^* is negligible. (Note that in the proof of security, this would be one of the "bad" events in the attack which contributes to the scheme's insecurity function, which we could bound, as we will see in the proof of Theorem 1).

[*Type-2 Outside Attacker.*] The type-2 outside attacker still does not have the receiver's private key but does *have the token* that the sender used to encrypt messages. Namely, the type-2 outside attacker is the "semi-trusted" third party that was appointed by the sender to release the token. However, we assume that this party can behave maliciously to break the confidentiality of the TCPKE ciphertexts. Like the type-1 outside attackers, the party can query token-ciphertext pairs of its choice to the decryption oracle of TCPKE. An attack game for a security notion associated with the type-2 attacker, which we call, "IND-TCPKE-T2CCA", is almost identical to that of IND-TCPKE-T1CCA except that the type-2 attacker cannot query the target token-ciphertext pair (tk^*, C^*) to the decryption oracle in Phase 4. (Since the type-2 attacker knows the target token, it is unreasonable to allow it).

Security against Inside Attacker: IND-TCPKE-ISCPA. As mentioned earlier, an "inside" attacker is assumed to possess the private key of the receiver but not the associated token. In other words, the inside attacker is the receiver itself. Having access to the token-embedded encryption oracle, the goal of the inside attacker is to defeat the confidentiality of a ciphertext created under the token. The security against this attack implies that the receiver cannot get any information about the plaintext message without getting the associated token.

We formalize the security notion for TCPKE against the inside attacker, which we call "IND-TCPKE-ISCPA". ("ISCPA" is read as "inside chosen plaintext attack").

Definition 3 (IND-TCPKE-ISCPA). Let A be an attacker whose running time is bounded by t which is polynomial in a security parameter k. We now consider the following game:

Phase 1: The key generation algorithm $\mathsf{GK}^{\mathsf{TCPKE}}(k)$ is run and a private and public key pair (sk, pk) is generated. Then, the token generation algorithm $\mathsf{GT}^{\mathsf{TCPKE}}(k)$ is run and a target token tk^* is generated. The *private/public* key pair (sk, pk) is given to A while tk^* is kept secret from A.
Phase 2: A queries a number of plaintexts, each of which is denoted by M, to the token-embedded encryption oracle and obtains a corresponding ciphertext $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M)$.
Phase 3: A outputs two equal-length plaintexts (M_0, M_1). $\beta \in \{0,1\}$ is then chosen at random and a target ciphertext $C^* = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M_\beta)$ is created and returned to A.
Phase 4: A queries a number of plaintexts to the token-embedded encryption oracle as in Phase 2.
Phase 5: A outputs his guess $\tilde{\beta} \in \{0,1\}$.

We define A's success by the probability $\mathbf{Succ}^{\mathrm{IND-TCPKE-ISCPA}}_{\mathsf{TPKE},\mathsf{A}}(k) = 2\Pr[\tilde{\beta} = \beta] - 1$. The TCPKE scheme is said to be IND-TCPKE-ISCPA secure if $\mathbf{Succ}^{\mathrm{IND-TCPKE-ISCPA}}_{\mathsf{TPKE},\mathsf{A}}(k)$ is negligible in k.

Relations among IND-TCPKE-T1CCA/T2CCA and IND-CCA. Once a token is revealed, the scheme TCPKE can be treated as a normal public key encryption scheme. We call this scheme "$\mathsf{PKE}_{\mathsf{TCPKE}}$", which can be defined as follows.

Definition 4 ($\mathsf{PKE}_{\mathsf{TCPKE}}$). A (normal) public key encryption scheme $\mathsf{PKE}_{\mathsf{TCPKE}}$ derived from TCPKE consists of the following algorithms.

– A Randomized Key Generation Algorithm $\mathsf{GK}^{\mathsf{PKE}}(k)$: This algorithm first computes $(sk, pk) = \mathsf{GK}^{\mathsf{TCPKE}}(k)$ and $tk = \mathsf{GT}^{\mathsf{TCPKE}}(k, pk)$, where $k \in \mathbf{N}$ is a security parameter. It then returns a private key $\overline{sk} = sk$ and a public key $\overline{pk} = pk||tk$.
– A Randomized Encryption Algorithm $\mathsf{E}^{\mathsf{PKE}}(\overline{pk}, M)$: This algorithm computes $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, M)$ and returns a ciphertext $\overline{C} = C||tk$.
– A Decryption Algorithm $\mathsf{D}^{\mathsf{PKE}}(\overline{sk}, \overline{C})$: This algorithm first parses $\overline{C}$ as $C||tk$ and computes $D = \mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$. It then returns D. (Note that D is either a plaintext $M \in \mathcal{P}$ or a special symbol "*Reject*").

Now we present a reduction result regarding the relationship between IND-TCPKE-T1CCA of TCPKE and IND-CCA of $\mathsf{PKE}_{\mathsf{TCPKE}}$. (Note that the formal definition of IND-CCA of public key encryption in general can be found in usual cryptographic literature including [1]). The following theorem implies that the scheme $\mathsf{PKE}_{\mathsf{TCPKE}}$ is secure in the IND-CCA sense and the token is chosen from an exponentially large space, then TCPKE is IND-TCPKE-T1CCA secure.

Theorem 1. *Suppose that a token for the* TCPKE *scheme is uniformly chosen at random from a space* $\mathcal{T}$ *such that* $|\mathcal{T}| > 2^{l_T(k)}$ *where* $l_T : \mathbb{N} \rightarrow \mathbb{N}$ *denotes linear function determining the length of a token. Assume that an attacker* A *making* q_{TE} *queries to the token-embedded encryption oracle and* q_D *queries to the decryption oracle defeats the IND-TCPKE-T1CCA of the* TCPKE *scheme within time* t*. Then for the attacker* A*, there exists an IND-CCA attacker* B *for the* $\mathsf{PKE}_{\mathsf{TCPKE}}$ *scheme, such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE,A}}(k) \leq \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k) + \frac{q_D}{2^{l_T(k)-1}}$$

and $t' = t + q_{TE}T_E$ *and* $q'_D = q_D$*, where* t'*,* T_E*, and* q'_D *denote the running time of* B*, the time required to encrypt a message using* $\mathsf{PKE}_{\mathsf{TCPKE}}$*, and the number of decryption oracle queries made by* B *respectively.*

The proof is given in Appendix A.

Regarding the relationship between IND-TCPKE-T2CCA and IND-CCA, we obtain a similar result. That is, if the scheme $\mathsf{PKE}_{\mathsf{TCPKE}}$ is secure in the IND-CCA sense then TCPKE is IND-TCPKE-T2CCA secure, which can be stated as follows.

Theorem 2. *Assume that an attacker* A *making* q_D *queries to the decryption oracle defeats the IND-TCPKE-T2CCA of the* TCPKE *scheme within time* t*. Then for the attacker* A*, there exists an IND-CCA attacker* B *for the* $\mathsf{PKE}_{\mathsf{TCPKE}}$ *scheme, such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}^{\mathrm{IND-TCPKE-T2CCA}}_{\mathsf{TCPKE,A}}(k) \leq \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k)$$

and $t' = t$ *and* $q'_D = q_D$*, where* t' *and* q'_D *denote the running time of* B *and the number of decryption oracle queries made by* B *respectively.*

Intuitively, what the type-2 outside attacker can do to break the IND-TCPKE-T2CCA of the TCPKE scheme is equivalent to what an IND-CCA attacker for the $\mathsf{PKE}_{\mathsf{TCPKE}}$ scheme can do in that the type-2 outside attacker "sees" the target token used to create a target ciphertext, which are all available to the IND-CCA attacker for the $\mathsf{PKE}_{\mathsf{TCPKE}}$ scheme.

4 Realization of Token-Controlled Public Key Encryption

Design Principles. We note that our TCPKE schemes will have a property called "public checkability." In publicly checkable encryption [1, 13], the ciphertext validity check for chosen ciphertext security can be performed by anyone, even who does not possess the decryption key. Hence, publicly checkable encryption schemes are useful in situations where a large number of incoming ciphertexts need to be checked and screened without having them to be decrypted as observed in [1]. Although this property is not a formal requirement of TCPKE as defined in Definition 1, it will also be useful in the situation of the token-controlled public key encryption, especially when the receivers do not have tokens to decrypt incoming ciphertexts yet, but want the validity of them to be checked. More precisely, the $\mathsf{PKE}_{\mathsf{TCPKE}}$ schemes derived from our TCPKE schemes will be publicly checkable without requiring the "token part" of the $\mathsf{PKE}_{\mathsf{TCPKE}}$ ciphertext, i.e. "tk" in $\overline{C} = C||tk$ (as described in Definition 4), to be involved in the validity checking for ensuring IND-CCA. (In other words, if C has checked before the token is released, the $\overline{C} = C||tk$ does not require to be checked again in the future).

However, care must be taken when constructing such schemes. Consider the following TCPKE scheme, which is very similar to our second scheme, but turns out to be insecure in the IND-TCPKE-T2CCA sense: Suppose that a plaintext M is encrypted to yield a ciphertext C such that $C = (u, v, h, s) = (g^r, M \oplus H_1(u, y^r) \oplus H_2(\tau), H_3(v, g^z), z - hr)$, where g is a generator of a group $\mathcal{G}$ of a prime order q; $r \in \mathbb{Z}_q^*$ and $z \in \mathbb{Z}_q^*$ are chosen at random; τ is a token chosen at random from an appropriate space. Suppose that τ is given to the (malicious) third party. Suppose also that C is a target ciphertext and τ is a target token. By the rule of the attack game of IND-TCPKE-T2CCA, the party cannot issue (τ, C) as a decryption oracle query. But he can replace τ with τ' and queries (τ', C) to the decryption oracle. Since the first part (u, v, h, s) is still valid, C will pass the validity test which checks whether $h = H_3(v, g^s u^h)$, and hence the oracle will return $M' = M \oplus H_2(\tau) \oplus H_2(\tau')$. However, the attacker can compute $H_2(\tau)$ and $H_2(\tau')$ by himself, so the M can easily be recovered from M'!

Another construction of a TCPKE scheme can be considered as follows. Let (pk, sk) be keys for an IND-CCA secure public-key encryption scheme with encryption algorithm E, and let E' be an encryption algorithm for an IND-CCA secure symmetric-key scheme. Then to encrypt a message m, choose a random key k as a token and send the ciphertext $E_{pk}(E'_k(m))$. However, to realize the IND-CCA secure symmetric-key scheme, one may need to employe the technique presented in [7], which makes a scheme somewhat complicated.

We now present our two TCPKE schemes.

Our First Scheme Based on Bilinear ElGamal Encryption + Short Signature. Our first scheme $\mathsf{TCPKE}_{\mathsf{BS}}$ is based on a combination of the bilinear pairing version of the ElGamal encryption scheme [4] and the Short Signature scheme [5]. The term "bilinear pairing" used in this paper refers to the admissible bilinear map $\hat{e} : \mathcal{G} \rightarrow \mathcal{F}$ [4] where $\mathcal{G}$ and $\mathcal{F}$ are groups of order prime q, which has the following property: 1) Bilinear: $\hat{e}(aR_1, bR_2) = \hat{e}(R_1, R_2)^{ab}$, where $R_1, R_2 \in \mathcal{G}$ and $a, b \in \mathbb{Z}_q^*$; 2) Non-degenerate: $\hat{e}$ does not send all pairs of points in $\mathcal{G} \times \mathcal{G}$ to the identity in $\mathcal{F}$. (Hence, if R is a generator of $\mathcal{G}$ then $\hat{e}(R, R)$ is a generator of $\mathcal{F}$).; 3)Computable: For all $R_1, R_2 \in \mathcal{G}$, the map $\hat{e}(R_1, R_2)$ is efficiently computable.

We use the Short Signature scheme proposed in [5] to convert the bilinear pairing version of the ElGamal encryption scheme to a publicly checkable IND-CCA secure scheme.
[*Description.*] The single-receiver token controlled public key encryption scheme $\mathsf{TCPKE}_{\mathsf{BS}}$ consists of the following algorithms:

– Randomized Key Generation Algorithm $\mathsf{GK}_{\mathsf{BS}}^{\mathsf{TCPKE}}(k)$: Choose two groups $\mathcal{G} = \langle P \rangle$ and $\mathcal{F}$ of the same prime order $q \geq 2^{l_q(k)}$ and chooses a generator P of $\mathcal{G}$. Then, construct a bilinear pairing $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$ and choose hash functions $\mathsf{H}_1 : \mathcal{F} \times \mathcal{G} \rightarrow \{0,1\}^{l_M(k)}$ and $\mathsf{H}_2 : \mathcal{G}^* \times \{0,1\}^{l_M(k)} \rightarrow \mathcal{G}^*$. (Note that $l_q : \mathbb{N} \rightarrow \mathbb{N}$ and $l_M : \mathbb{N} \rightarrow \mathbb{N}$ denote linear functions determining the length of q and message M respectively). Then, choose $x \in \mathbb{Z}_q^*$ uniformly at random and compute $Y = xP$. Return a public key $pk = (\mathrm{k}, \mathcal{G}, \hat{e}, q, P, Y, H_1, H_2, l_q, l_M)$ and a private key $sk = (\mathcal{G}, \hat{e}, q, P, x, H_1, H_2, k, l_q, l_M, d_{\mathcal{T}})$, where $d_{\mathcal{T}}$ denotes a description of a token space $\mathcal{T}$.
– Randomized Token Generation Algorithm $\mathsf{GT}_{\mathsf{BS}}^{\mathsf{TCPKE}}(k)$: Choose $T \in \mathcal{G}^* (= \mathcal{T})$ uniformly at random and return a token $tk = T$.
– Randomized Encryption Algorithm $\mathsf{E}_{\mathsf{BS}}^{\mathsf{TCPKE}}(pk, tk, M)$: Choose $r \in \mathbb{Z}_q^*$ uniformly at random and subsequently compute $U = rP$, $\kappa = \hat{e}(T, Y)^r$, $K = H_1(\kappa, T)$, $V = K \oplus M$, $L = H_2(U, V)$, and $W = rL$. Return a ciphertext $C = (U, V, W)$.
– Decryption Algorithm $\mathsf{D}_{\mathsf{BS}}^{\mathsf{TCPKE}}(sk, tk, C)$: If $\hat{e}(P, W) = \hat{e}(U, H_2(U, V))$ return $V \oplus H_1(\hat{e}(T, U)^x, T)$, otherwise, return "*Reject*".

[*Security Analysis.*] *Security against Outside/Limited Inside Attackers.* As shown in Theorems 1 and 2, and IND-CCA of a public key encryption scheme $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ derived from $\mathsf{TCPKE}_{\mathsf{BS}}$ following Definition 4 implies the security of $\mathsf{TCPKE}_{\mathsf{BS}}$ against outside/limited inside attackers in the single receiver setting. Hence, it is important to prove that $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ is IND-CCA secure and token-reusable.

We now prove that the hardness of the Bilinear Diffie-Hellman (BDH) problem [4] (given $aP, bP, cP \in \mathcal{G}$, computes $\hat{e}(P,P)^{abc} \in \mathcal{F}$) is sufficient for the $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ scheme to be IND-CCA secure in the random oracle model [3].

Theorem 3. *Assume that an attacker* A *making* q_{H_1} *and* q_{H_2} *queries to the random oracles* H_1 *and* H_2*, and* q_D *oracle queries to the decryption oracle defeats the IND-CCA of the* $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ *scheme within time* t*. Then, for the attacker* A*, there exists an attacker* B *that breaks the BDH problem within time* t' *such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}_{\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}},\mathsf{A}}^{\mathrm{IND-CCA}}(k) \leq 2\mathbf{Succ}_{\mathcal{G},\mathsf{B}}^{\mathrm{BDH}}(k) + \frac{q_D}{2^{k-1}}$$

and $t' = t + q_D(3T_{BP} + O(1))$*, where* T_{BP} *denotes the time for computing the bilinear pairing.*

In terms of the security of $\mathsf{TCPKE}_{\mathsf{BS}}$ against inside attackers in the single and multiple receiver setting, we investigate IND-TCPKE-ISCPA and IND-MRTCPKE-ISCPA of $\mathsf{TCPKE}_{\mathsf{BS}}$.

We first review the “modified Generalized Bilinear Inversion (GBI)” problem presented in [2]. The mGBI problem refers to the computational problem in which an attacker, given a generator P of $\mathcal{G}$ and $h \in \mathcal{F}$, is to find a pair $S \in \mathcal{G}$ such that $\hat{e}(S, P) = h$.

We then prove that the hardness of mGBI problem implies IND-TCPKE-ISCPA of $\mathsf{TCPKE}_{\mathsf{BS}}$.

Theorem 4. *Assume that an attacker* A *making* q_{H_1} *and* q_{H_2} *queries to the random oracles* H_1 *and* H_2 *defeats the IND-TCPKE-ISCPA of the* $\mathsf{TCPKE}_{\mathsf{BS}}$ *scheme within time* t*. Then, for the attacker* A*, there exists an attacker* B' *that breaks the mGBI problem within time* t' *such that for any* $k \in \mathbb{N}$*,*

$$\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-TCPKE-ISCPA}}_{\mathsf{TCPKE}_{\mathsf{BS}},\mathsf{A}}(k) \leq \mathbf{Succ}^{\mathrm{mGBI}}_{\mathcal{G},\mathsf{B}}(k)$$

and $t' = t + O(k^3)$*.*

Our Second Scheme Based on ElGamal Encryption + Schnorr Signature. Our second scheme $\mathsf{TCPKE}_{\mathsf{ES}}$ modifies the combination of the normal ElGamal encryption scheme constructed using a subgroup of the multiplicative group $\mathbb{Z}_p^*$ with prime p and the Schnorr [11] signature scheme [1, 12, 13].
[*Description*] The single-receiver token controlled public key encryption scheme $\mathsf{TCPKE}_{\mathsf{ES}}$ consists of the following algorithms:

– Randomized Key Generation Algorithm $\mathsf{GK}^{\mathsf{TCPKE}}_{\mathsf{ES}}(k)$: Choose a finite cyclic subgroup $\mathcal{G} = \langle g \rangle$ of a multiplicative group $\mathbb{Z}_p^*$ with prime p such that $Ord_{\mathcal{G}}(g) = q$, where q is a prime such that $|q| > 2^{l_q(k)}$. Choose hash functions $H_1 : \mathcal{G} \times \mathcal{G} \times \{0,1\}^* \rightarrow \{0,1\}^{l_M(k)}$ and $H_2 : \{0,1\}^{l_M(k)} \times \mathcal{G} \rightarrow \mathbb{Z}_q^*$. (Note that $l_q : \mathbb{N} \rightarrow \mathbb{N}$ and $l_M : \mathbb{N} \rightarrow \mathbb{N}$ denote linear functions determining the length of q and message M respectively). Choose $x \in \mathbb{Z}_q^*$ uniformly at random and compute $y = g^x$. Then, output a private and public key pair (sk, pk) such that $sk = (k, \mathcal{G}, g, p, q, x, H_1, H_2, l_q, l_M)$ and $pk = (k, \mathcal{G}, g, p, q, y, H_1, H_2, l_q, l_M, d_{\mathcal{T}})$, where $d_{\mathcal{T}}$ denotes a description of a token space $\mathcal{T}$.
– Randomized Token Generation Algorithm $\mathsf{GT}^{\mathsf{TCPKE}}_{\mathsf{ES}}(k)$: Choose $\tau \in \{0,1\}^{l_T(k)}$ uniformly at random and return a token $tk = \tau$. (Note that $l_T : \mathbb{N} \rightarrow \mathbb{N}$ denotes a linear function determining the length of a token).
– Randomized Encryption Algorithm $\mathsf{E}^{\mathsf{TCPKE}}_{\mathsf{ES}}(pk, tk, M)$: Choose $r \in \mathbb{Z}_q^*$ at random. Compute $u = g^r$, $\kappa = y^r$, $K = H_1(u, \kappa, \tau)$, and $v = K \oplus M$. Choose $z \in \mathbb{Z}_q^*$ at random. Compute $w = g^z$, $h = H_2(v, w)$, and $s = z - hr$. Return a ciphertext $C = (u, v, h, s)$.
– Decryption Algorithm $\mathsf{D}^{\mathsf{TCPKE}}_{\mathsf{ES}}(sk, tk, C)$: If $h = H_2(v, g^s u^h)$, compute $\kappa = y^x$ and $K = H_1(u, \kappa, \tau)$, and return $M = K \oplus v$, otherwise, return “*Reject*”.

[*Security Analysis.*] In the same way we analyzed the security of $\mathsf{TCPKE}_{\mathsf{BS}}$ against outside/limited inside attackers in the single receiver setting, we show that the public key encryption scheme $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{ES}}}$ derived from $\mathsf{TCPKE}_{\mathsf{ES}}$ following Definition 4 is IND-CCA secure.

We now prove that the intractability of the Gap Diffie-Hellman (GDH) problem [9], which is to solve the Computational Diffie-Hellman (CDH) problem with the help of Decisional Diffie-Hellman (DDH) oracle, is sufficient for the $\mathsf{PKE}_{\mathsf{ES}}$ scheme to be IND-CCA secure in the random oracle model.

Theorem 5. *Assume that an attacker* A *making* q_{H_1} *and* q_{H_2} *queries to the random oracles* H_1 *and* H_2*, and* q_D *oracle queries to the decryption oracle defeats the IND-CCA of the* $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{ES}}}$ *scheme within time* t*. Then, for the attacker* A*, there exists an attacker* B *that breaks the GDH problem within time* t' *such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{ES}}},\mathsf{A}}(k) \leq 2\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{G}}(k) + \frac{q_D}{2^{k-1}}$$

and $t' = t + q_D(3T_{DDH} + O(1))$*, where* T_{DDH} *denotes the running time of the DDH oracle.*

Security against Inside Attackers. IND-TCPKE-ISCPA and IND-TCPKE-ISCPA of the $\mathsf{TCPKE}_{\mathsf{ES}}$ scheme depends solely on the assumption that the token is chosen uniformly at random from the space $\{0,1\}^{l_T(k)}$ and the hash function H_1 is a random oracle. The advantages of the attackers for those notions are abounded by $O(\frac{1}{2^{l_T(k)}})$, which will be negligible if $l_T(k)$ is large.

References

1. M. Abe, *Combiniang Encryption and Proof of Knowledge in the Random Oracle Model*, Computer Journal 47(1), pp. 58–70, Oxford Uni. Press, 2004.
2. J. Baek and Y. Zheng, *Identity-based threshold signature scheme from the bilnear parings*, In IAS track of ITCC '04, pp. 124-128, IEEE Computer Society, 2004.
3. M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In ACM-CCCS '93, pp. 62–73, 1993.
4. D. Boneh and M. Franklin, *Identity-Based Encryption from he Weil Pairing*, In Crypto '01, LNCS 2139, pp. 213–229, Springer-Verlag, 2001.
5. D. Boneh, B. Lynn and H. Shacham, *Short Signatures from the Weil Pairing*, In Asiacrypt '01, LNCS 2248, pp. 566–582, Springer-Verlag, 2001.
6. G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan, *Conditional Oblivious Transfer and Timed-Release Encryption* , In Eurocrypt '99, LNCS 1592, pp. 74–89, Springer-Verlag, 1999.
7. A. Desai, *New Paradigms for Constructing Symmetric Encryption Schemes Secure against Chosen-Ciphertext Attack*, In Crypto '01, LNCS 1880, pp. 394–412, Springer-Verlag, 2000.
8. T. May, *Timed-Release Crypto*, Manuscript, 1993.
9. T. Okamoto and D. Pointcheval, *The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes*, In PKC '01, LNCS 1992, pp. 104–118, Springer-Verlag, 2001.
10. R. Rivest, A Shamir, and D. Wagner, *Time-Lock Puzzles and Timed-Release Crypto*, Technical Report, MIT/LCS/TR-684.
11. C. P. Schnorr, *Efficient Signature Generation for Smarts Cards*, Journal of Cryptology, Vol. 4, pp. 239–252, Springer-Verlag, 1991.

12. C. P. Schnorr and M. Jakobsson, *Security of Signed ElGamal Encryption* , In Asiacrypt '00, LNCS 1976, pp. 73–89, Springer-Verlag, 2000.
13. Y. Tsiounis and M. Yung: *On the Security of ElGamal-Based Encryption*, In PKC '98, LNCS 1431, pp. 117–134, Springer-Verlag, 1998.

A Proof of Theorem 1

Proof. Suppose that B is given a private/public key pair $(\overline{sk}, \overline{pk})$ such that $\overline{sk} = sk$ and $\overline{pk} = pk||tk$, where $(sk, pk) = \mathsf{GK}^{\mathsf{TCPKE}}(k)$ and $tk = \mathsf{GT}^{\mathsf{TCPKE}}(k)$. B lets $tk^* = tk$. then gives pk to A while keeps sk and tk^* secret. B then simulates the oracles that A has access to in the real attack as follows.

- Simulation of Token-Embedded Encryption Oracle: For each query M,
  - compute $\overline{C} = C||tk^* = \mathsf{E}^{\mathsf{PKE}}(\overline{pk}, M)$ and return C. (Note that $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk, M)$).
- Simulation of Encryption Oracle: For two equal-length plaintexts M_0 and M_1 as a challenge,
  - forward (M_0, M_1) to the encryption oracle of $\mathsf{PKE}_{\mathsf{TCPKE}}$ to obtain a (target) ciphertext $\overline{C^*} = C^*||tk^* = \mathsf{E}^{\mathsf{PKE}}(\overline{pk}, M_\beta)$ for random $\beta \in \{0,1\}$ and return C^*. (Note that $C^* = \mathsf{E}(pk, tk^*, M_\beta)$).
- Simulation of Decryption Oracle: For each query (C, tk),
  - if $(C, tk) = (C^*, tk^*)$, stop the simulation;
  - otherwise, forward $\overline{C} = C||tk$ to the decryption oracle of $\mathsf{PKE}_{\mathsf{TCPKE}}$ to obtain a decryption $D = \mathsf{D}^{\mathsf{PKE}}(\overline{sk}, \overline{C})$ and return D. (Note that $\mathsf{D}^{\mathsf{PKE}}(\overline{sk}, \overline{C}) = \mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$).
- When A outputs a guess $\tilde{\beta} \in \{0,1\}$ for the bit β, B returns $\tilde{\beta}$ as its guess.

Analysis. By the specifications presented above, the oracles that A has access to are perfectly simulated except for the case that A queries (C^*, tk^*) to the decryption oracle. Let TKBrk be an event that A queries (C, tk) such that $(C, tk) = (C^*, tk^*)$ in Phase 4. Since B's guess is exactly the same as A's guess $\tilde{\beta}$, if TKBrk does not happen, we have

$$\Pr[\tilde{\beta} = \beta | \neg \mathsf{TKBrk}] \le \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k).$$

Thus, we obtain the following:

$$\begin{aligned}\frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k) &= \Pr[\tilde{\beta} = \beta] \le \Pr[\tilde{\beta} = \beta | \neg \mathsf{TKBrk}] + \Pr[\mathsf{TKBrk}] \\ &\le \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k) + \Pr[\mathsf{TKBrk}].\end{aligned}$$

Since we have assumed that the token tk has been chosen uniformly at random from the space $\mathcal{T}$ such that $|\mathcal{T}| > 2^{l_T(k)}$ and the total q_D decryption oracle queries are made by A, $\Pr[\mathsf{TKBrk}] \le q_D/2^{l_T(k)}$ and hence we get the bound in the theorem statement. Note that the running time of B is $t' = t + q_{TE}T_E$ where T_E denotes the time required to encrypt a message using $\mathsf{PKE}_{\mathsf{TCPKE}}$. The number of decryption oracle queries made by B is $q'_D = q_D$. □