Timed-Ephemerizer: Make Assured Data Appear and Disappear

Qiang Tang

DIES, Faculty of EEMCS, University of Twente, the Netherlands
q.tang@utwente.nl

Abstract. The concept of Ephemerizer, proposed by Perlman, is a mechanism for assured data deletion. Ephemerizer provides a useful service that expired data deleted from the persistent storage devices will be unrecoverable, even if later on some of the private keys in the system are compromised. However, no security model has ever been proposed for this primitive and existing protocols have not been studied formally. In practice, a potential shortcoming of existing Ephemerizer protocols is that they are supposed to provide only assured deletion but not assured initial disclosure. In other words, there is no guarantee on when the data will be initially disclosed. In this paper, we formalize the notion of Timed-Ephemerizer which can be regarded as augmented Ephemerizer and can provide both assured initial disclosure and deletion for sensitive data. We propose a new Timed-Ephemerizer protocol and prove its security in the proposed security model.

1 Introduction

Rapid growth of information technology has greatly facilitated individuals and enterprizes to generate and store information (business transaction details, electronic health records, personal profiles, etc.). It is common that backups of the same piece of data will be placed on many different persistent storage devices, such as hard disks, tapes, and USB tokens. To protect the confidentiality, sensitive data are often firstly encrypted then stored on various devices, while the cryptographic keys also need to be stored and backuped on some persistent storage devices. With respect to storing data in persistent storage devices, there are two concerns.

1. It is relatively easy to recover data from persistent storage devices, even when the data has been deleted. As such, the US government specification has suggested to overwrite non-classified information three times [10].
2. Backups of encrypted sensitive data and cryptographic keys often reside in many devices. Consequently, it is difficult to make sure that all relevant backups have been deleted.

The above observations imply that an adversary may simultaneously obtain a copy of encrypted data and relevant cryptographic keys due to the potential management carelessness. Especially, this may be fairly easy for a malicious insider in organizations. As a result, even with encryption implemented, sensitive data may still be in potential danger.

To protect sensitive data from illegitimate leakage, Ephemerizer, proposed by Perlman [12,13], has shown a promising direction. For an application with Ephemerizer, data is encrypted using the public keys from both the data consumer and the Ephemerizer, and the ciphertext resides in the data consumer's persistent storage devices. If the data consumer wants to recover the data, it can decrypt the ciphertext with the help from the Ephemerizer. If we assume that the plaintext data will only reside in volatile storages[1] and the Ephemerizer will securely delete the expired ephemeral keys periodically, the decryption can only occur before the expiration of the relevant key pair of the Ephemerizer. In other words, a Ephemerizer protocol will provide assured deletion for sensitive data.

Contribution. In the literature, no security model has ever been proposed for Ephemerizer and existing protocols have not been analyzed formally. Some protocols have been shown suffering from security vulnerabilities (as surveyed in Section 2). In addition, we show that the Ephemerizer protocol in [9] suffers from serious security vulnerabilities in Appendix B. In practice, a potential shortcoming of existing Ephemerizer protocols is that they are only supposed to provide assured deletion but not assured initial disclosure. In other words, there is no guarantee on when the data will be initially disclosed.

We formalize the notion of Timed-Ephemerizer, aimed to provide an assured disclosure policy enforcement for the lifecycle of sensitive data, where the lifecycle is marked by the initial disclosure and the deletion after expiration. Conceptually, the new primitive can be regarded as augmented Ephemerizer by combining Ephemerizer (for assured deletion)[12,13] and Timed-Release Encryption (for assured initial disclosure)[8]. In other words, Ephemerizer can be seen as a Timed-Ephemerizer without the timed re-lease property. Furthermore, we propose a new Timed-Ephemerizer protocol and prove its security in our model.

Organization. The rest of the paper is organized as follows. In Section 2 we briefly review the relevant works on Ephemerizer and Timed-Release Encryption. In Section 3 we introduce the concept of Timed-Ephemerizer and formalize the security properties. In Section 4 we propose a new Timed-Ephemerizer protocol and prove its security. In Section 5 we conclude the paper.

[1] In contrast to persistent storage devices, it is more difficult for an adversary to corrupt volatile storage devices (for example, most forms of modern random access memory) because the data in such devices will disappear when the electricity/power is gone. However, it is worth noting that this could be very subtle in the presence of side channel attacks, especially when considering the cold boot attacks [6].

2 Related Work

2.1 Ephemerizer Protocols

Perlman [12,13] proposes two Ephemerizer protocols without providing rigorous security proofs. One protocol uses a blind encryption technique, which is a kind of homomorphic property between two encryption schemes. The other protocol uses a triple encryption technique, where data is encrypted using a symmetric key which is sequentially encrypted using the public key of the data consumer, the public key of the Ephemerizer, and the public key of the data consumer. However, this protocol has been shown suffering from a fatal vulnerability by Nair *et al.* [9]. In addition, Nair *et al.* [9] observe that both protocols proposed by Perlman do not provide support for fine-grained user settings on the lifetime of the data. As a solution, Nair *et al.* propose a protocol using identity-based public-key encryption. However, they have not provided a security analysis in a formal security model. In Appendix B, we show that their protocol also suffers from fatal vulnerabilities.

2.2 Timed-Release Encryption

The concept of Timed-Release Encryption (TRE), i.e. sending a message which can only be decrypted after a pre-defined release time, is attributed to May [8]. Later on, Rivest, Shamir, and Wagner further elaborate on this concept and gave a number of its applications including electronic auctions, key escrow, chess moves, release of documents over time, payment schedules, press releases [14]. Hwang, Yum, and Lee [7] extend the concept of TRE schemes to include the Pre-Open Capability which allows the message sender to assist the receiver to decrypt the ciphertext before the pre-defined disclosure time. Later on, Dent and Tang [5] propose a refined model and comprehensive analysis for this extended primitive.

There are two approaches to embed a timestamp in a ciphertext. One approach, proposed in [14], is that a secret is transformed in such a way that all kinds of machines (serial or parallel) take at least a certain amount of time to solve the underlying computational problems (puzzle) in order to recover the secret. The release time is equal to the time at which the puzzle is released plus the minimum amount of time that it would take to solve the puzzle. However, this means that not all users are capable of decrypting the ciphertext at the release time as they may have different computing power. The other approach is to use a trusted time server, which, at an appointed time, will assist in releasing a secret to help decrypt the ciphertext (e.g. [3,14]). Using this approach, the underlying schemes require interaction between the server and the users, and should prevent possible malicious behaviour of the time server. In this paper, we will adopt the second approach because, regardless of the computing power of all involved entities, it can provide assured disclosure time under appropriate assumptions.

3 The Concept of Timed-Ephemerizer

Informally, a Timed-Ephemerizer protocol guarantees that data will only be available during a pre-defined lifecycle, beyond which no adversary can recover the data even if it has compromised all existing private keys in the system. Compared with Ephemerizer protocols [9,12,13], a Timed-Ephemerizer protocol explicitly provides the guarantee that data can only be available after the pre-defined initial disclosure time.

3.1 The Algorithm Definitions

Generally, a Timed-Ephemerizer protocol involves the following types of entities: time server, data generator, data consumer, and Ephemerizer.

- Time server, which will publish timestamps periodically. We assume that the time server acts properly in generating its parameters and publishing the timestamps. However, concerning the privacy of data, we take into account the fact that the time server may be curious, i.e. it may try to decrypt the ciphertext.
- Data generator, which will make her data available to a data consumer. The data generator defines the lifecycle of her data.
- Data consumer, which will access the data generator's data. A data consumer could be curious in the way that it may try to access data before the initial disclosure time.
- Ephemerizer, which is trusted to publish and revoke ephemeral public/private key pairs periodically. However, the Ephemerizer could be curious in the sense that it may try to decrypt the ciphertext.

Remark 1. Compared with an Ephemerizer protocol, a Timed-Ephemerizer protocol has one additional entity, namely the time server. One may have the observation that the Ephemerizer can be required to release timestamps so that the time server can be eliminated. However, we argue that the separation of functionalities provides a higher level of security in general. First of all, the time server only needs to publish timestamps without any additional interaction with other entities. In practice, the risk that time server is compromised is less than that for the Ephemerizer. Secondly, the risk that both the Ephemerizer and the time server are compromised is less than that any of them is compromised.

A Timed-Ephemerizer protocol consists of the following polynomial-time algorithms. Let ℓ be the security parameter.

- $\mathsf{Setup}_T(\ell)$: Run by the time server, this algorithm generates a public/private key pair (PK_T, SK_T).
- $\mathsf{TimeExt}(t, SK_T)$: Run by the time server, this algorithm generates a timestamp TS_t. It is assumed that the time server publishes TS_t at the point t. Throughout the paper, the notation $t < t'$ means t is earlier than t'.

– $\mathsf{Setup}_E(\ell)$: Run by the Ephemerizer, this algorithm generates a set of tuples $(PK_{t_{eph_j}}, SK_{t_{eph_j}}, t_{eph_j})$ for $j \geq 1$, where $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ is a public/private key pair and t_{eph_j} is the expiration time of the key pair. The Ephemerizer will securely delete $SK_{t_{eph_j}}$ at the point t_{eph_j}. We assume that there is only one ephemeral key pair for any expiration time t_{eph_j}. In addition, we assume $t_{eph_j} < t_{eph_k}$ if $j < k$.
– $\mathsf{Setup}_U(\ell)$: Run by a data consumer, this algorithm generates a public/private key pair (PK_U, SK_U).
– $\mathsf{Generate}(M, t_{int}, PK_U, PK_{t_{eph_j}}, PK_T)$: Run by the data generator, this algorithm outputs a ciphertext C. For the message M, t_{int} is the initial disclosure time and t_{eph_j} is the expiration time. We explicitly assume that both (t_{int}, t_{eph_j}) and C should be sent to the data consumer.
– $\mathsf{Retrieve}(C, TS_{t_{int}}, SK_U; SK_{t_{eph_j}})$: Interactively run between the data consumer and the Ephemerizer, this algorithm outputs a plaintext M or an error symbol $\perp$ to the data consumer. We explicitly make the following assumption. The data consumer has $(C, TS_{t_{int}}, SK_U)$ as the input and sends t_{eph_j} to the Ephemerizer in advance, so that the Ephemerizer uses $SK_{t_{eph_j}}$ as the input for the upcoming algorithm execution.

Remark 2. In the algorithm definitions, besides the explicitly specified parameters, other public parameters could also be specified and be implicitly part of the input. We omit those parameters for the simplicity of description.

With a Timed-Ephemerizer protocol, the workflow is similar to that of an Ephemerizer protocol.

1. The data generator runs the algorithm Generate to encrypt her data. The difference is that this algorithm involves the public key of the time server.
2. The data consumer runs the algorithm Retrieve to decrypt the ciphertext with the help from the Ephemerizer. The difference is that this algorithm involves a timestamp from the time server.

3.2 The Security Definitions

We first describe some conventions for writing probabilistic algorithms and experiments. The notation $u \in_R S$ means u is randomly chosen from the set S. If $\mathcal{A}$ is a probabilistic algorithm, then $v \xleftarrow{\$} \mathcal{A}^{(\mathsf{f}_1, \mathsf{f}_2, \cdots)}(x, y, \cdots)$ means that v is the result of running $\mathcal{A}$, which takes $x, y, \cdots$ as input and has any polynomial number of oracle queries to the functions $\mathsf{f}_1, \mathsf{f}_2, \cdots$. As a standard practice, the security of a protocol is evaluated by an experiment between an attacker and a challenger, where the challenger simulates the protocol executions and answers the attacker's oracle queries. Without specification, algorithms are always assumed to be polynomial-time.

A Timed-Ephemerizer protocol is aimed to guarantee that data will only be available during its lifecycle, while neither before the initial disclosure time nor after the expiration time. We assume that the validation of public keys in the protocol can be verified by all the participants. Nonetheless, we generally assume that an outside adversary is active, which means that the adversary may compromise the protocol participants and fully control the communication channels (i.e. capable of deleting, relaying, and replacing the messages exchanged between the participants). Considering the threats against confidentiality, we identify three categories of adversaries.

- Type-I adversary: This type of adversary wants to access data before its initial disclosure time. Type-I adversary represents a curious data consumer and also a malicious outside entity which has compromised the Ephemerizer and the data consumer before the initial disclosure time of the data.
- Type-II adversary: This type of adversary wants to access data after its expiration time. Type-II adversary represents a malicious outside entity which has compromised the time server, the Ephemerizer, and the data consumer after the expiration time of the data.
- Type-III adversary: This type of adversary represents a curious time server and a curious Ephemerizer, and also a malicious outside entity which has compromised the Ephemerizer and the data consumer

The implications of a Type-I adversary and a Type-II adversary are clear for a Timed-Ephemerizer protocol. Nonetheless, the existence of a Type-III adversary still makes sense even in the presence of these two types of adversary. Compared with a Type-I adversary, a Type-III adversary has the advantage of accessing the private key (and all timestamps) of the time server; while compared with a Type-II adversary, a Type-III adversary has the advantage of accessing all the private keys of the Ephemerizer. However, a Type-III adversary does not have direct access to the data consumer's private key.

Remark 3. It is worth stressing that when the adversary compromises an entity (the time server, the Ephemerizer, or the data consumer) it will obtain the private keys possessed by that entity. For example, if the Ephemerizer is compromised at the point t, then it will obtain all the private keys $SK_{t_{eph_j}}$ for $t_{eph_j} > t$. However, we do not take into account the compromise of ephemeral session secrets during the executions of algorithms.

Definition 1. *A Timed-Ephemerizer protocol achieves Type-I semantic security if any polynomial-time adversary has only a negligible advantage in the following semantic security game (as shown in Figure 1), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

In more detail, the attack game between the challenger and the adversary $\mathcal{A}$ performs as follows. In this game the challenger simulates the functionality of the time server.

1. The challenger runs Setup_T to generate (PK_T, SK_T), runs Setup_E to generate $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $j \geq 1$, and runs Setup_U to generate (PK_U, SK_U). Except for SK_T, all private keys and all public parameters are given to the adversary.

1. $(PK_T, SK_T) \xleftarrow{\$} \mathsf{Setup}_T(\ell)$; $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $j \geq 1 \xleftarrow{\$} \mathsf{Setup}_E(\ell)$; $(PK_U, SK_U) \xleftarrow{\$} \mathsf{Setup}_U(\ell)$
2. $(M_0, M_1, t^*_{int}, PK_{t_{eph_i}}) \xleftarrow{\$} \mathcal{A}^{(\mathsf{TimeExt})}(SK_{t_{eph_j}}$ for $j \geq 1, SK_U)$
3. $b \xleftarrow{\$} \{0,1\}; C_b \xleftarrow{\$} \mathsf{Generate}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T)$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{TimeExt})}(C_b, SK_{t_{eph_j}}$ for $j \geq 1, SK_U)$

Fig. 1. Semantic Security against Type-I Adversary

2. The adversary can adaptively query the TimeExt oracle, for which the adversary provides a time t and gets a timestamp TS_t from the challenger. At some point, the adversary sends the challenger two equal-length plaintext M_0, M_1 on which it wishes to be challenged, and two timestamps (t^*_{int}, t_{eph_i}). The only restriction is that the TimeExt oracle should not have been queried with $t \geq t^*_{int}$.
3. The challenger picks a random bit $b \in \{0, 1\}$ and gives the adversary C_b as the challenge, where

$$C_b = \mathsf{Generate}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T).$$

4. The adversary can continue to query the TimeExt oracle with the same restriction as in Step 2.
5. Eventually, the adversary outputs b'.

In the above attack game, the adversary is Type-I because it has access to SK_U and $SK_{t_{eph_j}}$ for any $j \geq 1$

Remark 4. The restriction in steps 2 and 4 of the above game, namely "the TimeExt oracle should not have been queried with $t \geq t^*_{int}$.", implies that the adversary tries to recover a message before the initial disclosure time. This coincides with the definition of Type-I adversary.

Definition 2. *A Timed-Ephemerizer protocol achieves Type-II semantic security if any polynomial time adversary has only a negligible advantage in the following semantic security game (as shown in Figure 2), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

1. $(PK_T, SK_T) \xleftarrow{\$} \mathsf{Setup}_T(\ell)$; $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $j \geq 1 \xleftarrow{\$} \mathsf{Setup}_E(\ell)$; $(PK_U, SK_U) \xleftarrow{\$} \mathsf{Setup}_U(\ell)$
2. $(M_0, M_1, t^*_{int}, PK_{t_{eph_i}}) \xleftarrow{\$} \mathcal{A}^{(\mathsf{Retrieve})}(SK_T, SK_{t_{eph_j}}$ for $j > i, SK_U)$
3. $b \xleftarrow{\$} \{0,1\}; C_b \xleftarrow{\$} \mathsf{Generate}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T)$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{Retrieve})}(C_b, SK_T, SK_{t_{eph_j}}$ for $j > i, SK_U)$

Fig. 2. Semantic Security against Type-II Adversary

In more detail, the attack game between the challenger and the adversary $\mathcal{A}$ performs as follows. In this game the challenger simulates the functionalities of both the Ephemerizer and the data consumer.

1. The challenger runs Setup_T to generate (PK_T, SK_T), runs Setup_E to generate $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $j \geq 1$, and runs Setup_U to generate (PK_U, SK_U). The private key SK_T and all public parameters are given to the adversary.
2. The adversary can adaptively issue the following two types of $\mathsf{Retrieve}$ oracle queries.
   (a) D-type $\mathsf{Retrieve}$ oracle query: In each oracle query, the adversary impersonates the Ephemerizer and provides (t_{int}, t_{eph_j}) and C to the challenger, which then uses $(C, TS_{t_{int}}, SK_U)$ as input and runs the $\mathsf{Retrieve}$ algorithm with the adversary to decrypt C by assuming that the initial disclosure time is t_{int} and the expiration time is t_{eph_j}.
   (b) E-type $\mathsf{Retrieve}$ query: In each oracle query, the adversary impersonates the data consumer to the Ephemerizer and sends t_{eph_j} to the challenger, which uses $SK_{t_{eph_j}}$ as the input and runs the $\mathsf{Retrieve}$ algorithm with the adversary.

   At some point, the adversary sends the challenger two equal-length plaintext M_0, M_1 on which it wishes to be challenged, and two timestamps (t^*_{int}, t_{eph_i}). In this phase, the adversary can query for SK_U and $SK_{t_{eph_j}}$ for any $j > i$ with the following restriction: if SK_U has been queried, then any E-type $\mathsf{Retrieve}$ oracle query with the input t_{eph_j} for any $j \leq i$ is forbidden.
3. The challenger picks a random bit $b \in \{0, 1\}$ and gives the adversary C_b as the challenge, where

$$C_b = \mathsf{Generate}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T).$$

4. The adversary can continue to issue oracle queries as in Step 2 with the same restriction.
5. The adversary $\mathcal{A}$ outputs b'.

In the above attack game, the adversary is Type-II because it has access to the private keys SK_T, SK_U, and $SK_{t_{eph_j}}$ for any $j > i$.

Remark 5. In the above game, the privilege, that the adversary can issue the two types of $\mathsf{Retrieve}$ oracle queries, reflects the fact that the adversary has complete control over the communication link between the data consumer and the Ephemerizer. In practice, such an adversary can initiate the $\mathsf{Retrieve}$ algorithm with both the Ephemerizer and the data consumer. The first case is modeled by the E-type $\mathsf{Retrieve}$ query, while the second case is modeled by the D-type $\mathsf{Retrieve}$ query.

Remark 6. The restriction in the above game, namely "if SK_U has been queried, then E-type $\mathsf{Retrieve}$ oracle query with the input t_{eph_j} for any $j \leq i$ is forbidden.", reflects the fact that the adversary tries to recover a message after its expiration time t_{eph_i} (when the ephemeral keys $SK_{t_{eph_j}}$ for any $j \leq i$ should have been securely deleted by the Ephemerizer). This coincides with the definition of Type-II adversary.

Definition 3. *A Timed-Ephemerizer protocol achieves Type-III semantic security if any polynomial time adversary has only a negligible advantage in the following semantic security game (as shown in Figure 3), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

```
1. $(PK_T, SK_T) \xleftarrow{\$} \mathsf{Setup}_T(\ell)$; $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $j \geq 1 \xleftarrow{\$} \mathsf{Setup}_E(\ell)$; $(PK_U, SK_U) \xleftarrow{\$} \mathsf{Setup}_U(\ell)$
2. $(M_0, M_1, t^*_{int}, PK_{t_{eph_i}}) \xleftarrow{\$} \mathcal{A}^{(\mathsf{Retrieve})}(SK_T, SK_{t_{eph_j}}$ for $j \geq 1)$
3. $b \xleftarrow{\$} \{0,1\}; C_b \xleftarrow{\$} \mathsf{Generate}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T)$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{Retrieve})}(C_b, SK_T, SK_{t_{eph_j}}$ for $j \geq 1)$
```

Fig. 3. Semantic Security against Type-III Adversary

In more detail, the attack game between the challenger and the adversary $\mathcal{A}$ performs as the following. In this game the challenger simulates the functionality of the data consumer.

1. The challenger runs Setup_T to generate (PK_T, SK_T), runs Setup_E to generate $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $j \geq 1$, and runs Setup_U to generate (PK_U, SK_U). The private key SK_T, all ephemeral private keys $SK_{t_{eph_j}}$ for $j \geq 1$, and all public parameters are given to the adversary.
2. The adversary can adaptively issue the D-type Retrieve oracle query (defined as above). At some point, the adversary sends the challenger two equal-length plaintext M_0, M_1 on which it wishes to be challenged, and two timestamps (t^*_{int}, t_{eph_i}).
3. The challenger picks a random bit $b \in \{0,1\}$ and gives the adversary C_b as the challenge, where
$$C_b = \mathsf{Generate}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T).$$
4. The adversary can continue to query the Retrieve oracle as in Step 2.
5. The adversary $\mathcal{A}$ outputs b'.

In the above attack game, the adversary is Type-III because it has access to the private keys SK_T and $SK_{t_{eph_j}}$ for any $j \geq 1$.

Remark 7. In the above game, expect for the data consumer's private key, the adversary is allowed to access all other secrets. In particular, this means that the adversary can compromise both the time server and the Ephemerizer at any time. This coincides with the definition of Type-III adversary.

4 A New Timed-Ephemerizer Protocol

4.1 Preliminary of Pairing

We review the necessary knowledge about pairing and the related assumptions. More detailed information can be found in the seminal paper [2]. A pairing (or, bilinear map) satisfies the following properties:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two multiplicative groups of prime order p;
2. g is a generator of $\mathbb{G}$;
3. $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is an efficiently-computable bilinear map with the following properties:
   - Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
   - Non-degenerate: $\hat{e}(g, g) \neq 1$.

The Bilinear Diffie-Hellman (BDH) problem in $\mathbb{G}$ is as follows: given a tuple $g, g^a, g^b, g^c \in \mathbb{G}$ as input, output $\hat{e}(g, g)^{abc} \in \mathbb{G}_1$. An algorithm $\mathcal{A}$ has advantage ϵ in solving BDH in $\mathbb{G}$ if

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = \hat{e}(g, g)^{abc}] \geq \epsilon.$$

Similarly, we say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the decision BDH problem in $\mathbb{G}$ if

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, T) = 0]| \geq \epsilon.$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits of $\mathcal{A}$.

Definition 4. *We say that the (decision) (t, ϵ)-BDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the (decision) BDH problem in $\mathbb{G}$.*

Besides these computational/decisional assumptions, the Knowledge of Exponent (KE) assumption is also used in a number of papers (e.g. [1,4]). The KE assumption is defined as follows.

Definition 5. *For any adversary $\mathcal{A}$, which takes a KE challenge (g, g^a) as input and returns (C, Y) where $Y = C^a$, there exists an extractor $\mathcal{A}'$, which takes the same input as $\mathcal{A}$ returns c such that $g^c = C$.*

4.2 The Proposed Construction

The philosophy behind the proposed protocol is similar to the blind encryption technique [12,13]. The data generator encrypts the data jointly using the ephemeral public key of the Ephemerizer and the public key of the time server, then the ciphertext is encrypted using the public key of the data consumer. The main difference (and advantage) is that we avoid using blind encryption technique while using an efficient re-randomization technique with the XOR ($\oplus$) operation.

Let ℓ be the security parameter and $\{0, 1\}^n$ be the message space of data consumer, where n is a polynomial in ℓ. The polynomial-time algorithms are defined as follows.

- $\mathsf{Setup}_T(\ell)$: This algorithm generates the following parameters: a multiplicative group $\mathbb{G}$ of prime order p, a generator g of $\mathbb{G}$, and a multiplicative group $\mathbb{G}_1$ of the same order as $\mathbb{G}$, a polynomial-time computable bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, a cryptographic hash function $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}$, and a long-term public/private key pair (PK_T, SK_T) where $SK_T \in_R \mathbb{Z}_p$ and $PK_T = g^{SK_T}$. The time server also publishes $(\mathbb{G}, \mathbb{G}_1, p, g, \hat{e}, \mathsf{H}_1)$. Suppose the time server possesses the identity ID_T.
- $\mathsf{TimeExt}(t, SK_T)$: This algorithm returns $TS_t = \mathsf{H}_1(ID_T \| t)^{SK_T}$.
- $\mathsf{Setup}_E(\ell)$: Suppose that the Ephemerizer possesses the identity ID_E. The Ephemerizer uses the same set of parameter $(\mathbb{G}, \mathbb{G}_1, p, g, \hat{e})$ as by the time server and selects the supported expiration times t_{eph_j} $(1 \le j \le N)$ where N is an integer. The Ephemerizer generates a master key pair $(PK_E^{(0)}, SK_E^{(0)})$, where $SK_E^{(0)} \in_R \mathbb{Z}_p$ and $PK_E^{(0)} = g^{SK_E^{(0)}}$, and two hash functions

  $$\mathsf{H}_2 : \{0,1\}^* \to \mathbb{G},\ \mathsf{H}_3 : \mathbb{G}_1 \to \{0,1\}^n,$$

  and sets, for $1 \le j \le N$,

  $$PK_{t_{eph_j}}^{(0)} = ID_E \| t_{eph_j},\ SK_{t_{eph_j}}^{(0)} = \mathsf{H}_2(ID_E \| t_{eph_j})^{SK_E^{(0)}}.$$

  The Ephemerizer generates another master key pair $(PK_E^{(1)}, SK_E^{(1)})$ for an identity-based public key encryption scheme $\mathcal{E}_1$ with the encryption/decryption algorithms $(\mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$, and generates the ephemeral key pairs $(PK_{t_{eph_j}}^{(1)},\ SK_{t_{eph_j}}^{(1)})$ for $1 \le j \le N$, where $PK_{t_{eph_j}}^{(1)} = ID_E \| t_{eph_j}$. Suppose the message space and ciphertext space of the encryption scheme $\mathcal{E}_1$ are $\mathcal{Y}$ and $\mathcal{W}$, respectively.
  The Ephemerizer keeps a set of tuples $(PK_{t_{eph_j}}, SK_{t_{eph_j}}, t_{eph_j})$ for $j \ge 1$, where

  $$PK_{t_{eph_j}} = (PK_{t_{eph_j}}^{(0)}, PK_{t_{eph_j}}^{(1)}),\ SK_{t_{eph_j}} = (SK_{t_{eph_j}}^{(0)}, SK_{t_{eph_j}}^{(1)})$$

  The Ephemerizer publishes the long-term public keys $PK_E^{(0)}, PK_E^{(1)}$.
- $\mathsf{Setup}_U(\ell)$: This algorithm generates a public/private key pair (PK_U, SK_U) for a public key encryption scheme $\mathcal{E}_2$ with the encryption/decryption algorithms $(\mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$. Suppose the message space of $\mathcal{E}_2$ is $\mathcal{X}$ and the ciphertext space is $\mathcal{D}$. The data consumer publishes the following hash functions $\mathsf{H}_4, \mathsf{H}_5, \mathsf{H}_6, \mathsf{H}_7, \mathsf{H}_8, \mathsf{H}_9$.

  $$\mathsf{H}_4 : \mathbb{G} \times \mathbb{G} \to \mathbb{G},\ \mathsf{H}_5 : \mathcal{X} \to \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n,$$

  $$\mathsf{H}_6 : \mathcal{X} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n \times \mathcal{D} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n \to \{0,1\}^n,$$

  $$\mathsf{H}_7 : \mathcal{Y} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n \times \mathcal{W} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n \to \{0,1\}^n,$$

  $$\mathsf{H}_8 : \mathcal{Y} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n \to \{0,1\}^n,\ \mathsf{H}_9 : \mathcal{Y} \to \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n.$$

– $\mathsf{Generate}(M, t_{int}, PK_U, PK_{t_{eph_j}}, PK_T)$: This algorithm outputs a ciphertext C, where

$$r_1, r_2 \in_R \mathbb{Z}_p,\ X \in_R \mathcal{X},\ C_1 = g^{r_1},\ C_2 = g^{r_2},\ C_3 = \mathsf{H}_4(C_1 \| C_2)^{r_1},$$

$$\begin{aligned} C_4 &= M \oplus \mathsf{H}_3(\hat{e}(\mathsf{H}_2(PK^{(0)}_{t_{eph_j}}), PK^{(0)}_E)^{r_1} \cdot \hat{e}(\mathsf{H}_1(ID_T \| t_{int}), PK_T)^{r_2}) \\ &= M \oplus \mathsf{H}_3(\hat{e}(\mathsf{H}_2(ID_E \| t_{eph_j}), C_1)^{SK^{(0)}_E} \cdot \hat{e}(\mathsf{H}_1(ID_T \| t_{int}), C_2)^{SK_T}), \end{aligned}$$

$$C_5 = \mathsf{Encrypt}_2(X, PK_U),\ C_6 = \mathsf{H}_5(X) \oplus (C_1 \| C_2 \| C_3 \| C_4),$$

$$C_7 = \mathsf{H}_6(X \| C_1 \| C_2 \| C_3 \| C_4 \| C_5 \| C_6),\ C = (C_5, C_6, C_7).$$

– $\mathsf{Retrieve}(C, TS_{t_{int}}, SK_U; SK_{t_{eph_j}})$:

1. The data consumer decrypts C_5 to obtain X, and aborts if the following inequation is true.

$$C_7 \neq \mathsf{H}_6(X \| (C_6 \oplus \mathsf{H}_5(X)) \| C_5 \| C_6)$$

   Otherwise it computes $C_1 \| C_2 \| C_3 \| C_4 = \mathsf{H}_5(X) \oplus C_6$. The data consumer then computes and sends $(C', TS_{t_{int}})$ to the Ephemerizer, where

$$M' \in_R \{0,1\}^n,\ C'_1 = C_1,\ C'_2 = C_2,\ C'_3 = C_3,\ C'_4 = M' \oplus C_4,$$

$$Y \in_R \mathcal{Y},\ C'_5 = \mathsf{Encrypt}_1(Y, PK^{(1)}_{t_{eph_j}}),\ C'_6 = \mathsf{H}_9(Y) \oplus (C'_1 \| C'_2 \| C'_3 \| C'_4),$$

$$C'_7 = \mathsf{H}_7(Y \| C'_1 \| C'_2 \| C'_3 \| C'_4 \| C'_5 \| C'_6),\ C' = (C'_5, C'_6, C'_7).$$

2. If the ephemeral key $SK_{t_{eph_j}} = (SK^{(0)}_{t_{eph_j}}, SK^{(1)}_{t_{eph_j}})$ has not expired, the Ephemerizer decrypts C'_5 to obtain Y, and aborts if

$$C'_7 \neq \mathsf{H}_7(Y \| (C'_6 \oplus \mathsf{H}_9(Y)) \| C'_5 \| C'_6)$$

   It then computes $C'_1 \| C'_2 \| C'_3 \| C'_4 = \mathsf{H}_9(Y) \oplus C'_6$, and aborts if

$$\hat{e}(C'_3, g) \neq \hat{e}(C'_1, \mathsf{H}_4(C'_1 \| C'_2))$$

   Finally, it sends C'' to the data consumer, where

$$\begin{aligned} C'' &= \mathsf{H}_8(Y \| C'_1 \| C'_2 \| C'_3 \| C'_4) \oplus C'_4 \oplus \mathsf{H}_3(\hat{e}(C'_1, SK^{(0)}_{t_{eph_j}}) \cdot \hat{e}(TS_{t_{int}}, C'_2)) \\ &= \mathsf{H}_8(Y \| C'_1 \| C'_2 \| C'_3 \| C'_4) \oplus M' \oplus M. \end{aligned}$$

3. The data consumer recovers $M = \mathsf{H}_8(Y \| C'_1 \| C'_2 \| C'_3 \| C'_4) \oplus M' \oplus C''$.

As in the case of the hybrid PKI-IBC protocol [9], the proposed protocol also adopts the concept of identity-based encryption [2,15]. As a result, the Ephemerizer avoids publishing a large volume of ephemeral public keys, which is however the case in [12,13]. Compared with the protocol in [9], the concrete difference is that the master private key $SK_E = (SK^{(0)}_E, SK^{(1)}_E)$ is only required to be ephemeral, i.e. after generating the ephemeral private keys, the Ephemerizer can delete SK_E.

Remark 8. In the execution of Retrieve, the timestamp $TS_{t_{int}}$ is a required input. Intuitively, before the time server publishes the timestamp, it is infeasible for the data consumer and the Ephemerizer to run Retrieve to recover the message. Lemma 1 in the next section formalizes this intuition.

Remark 9. For a Timed-Ephemerizer protocol, the semantic securities against Type-I and Type-III adversaries are relatively easy to achieve, given the existing timed-release encryption techniques. The difficulty lies in the semantic security against Type-II adversary, which fully controls the communication channel and is capable of adaptively compromising all parties in the system. In fact, this has resulted in the complexity of the above protocol.

4.3 The Security Analysis

The following three lemmas show that the proposed protocol is secure against all three types of adversaries. Their proofs are in the Appendix A.

Lemma 1. *The proposed scheme achieves semantic security against Type-I adversary based on the BDH assumption in the random oracle model.*

Lemma 2. *The proposed scheme achieves semantic security against Type-II adversary based on the BDH and the KE assumptions in the random oracle model given that the public key encryption schemes $\mathcal{E}_1$ and $\mathcal{E}_2$ are one-way permutation.*

Lemma 3. *The proposed scheme achieves semantic security against Type-III adversary in the random oracle model given that the public key encryption schemes $\mathcal{E}_1$ and $\mathcal{E}_2$ are one-way permutation.*

5 Conclusion

In this paper we revisited the concept of Ephemerizer proposed by Perlman, and formalized the notion of Timed-Ephemerizer, aimed to provide an assured lifecycle for sensitive data, and proposed a new Timed-Ephemerizer protocol and proved its security in the proposed security model. For this new concept of Timed-Ephemerizer, a number of interesting research questions remain open. We list two of them here. One is to investigate more efficient and secure protocols for Timed-Ephemerizer. Especially, note that the random oracle paradigm has been heavily used in the security analysis of the proposed protocol. It is interesting to design secure protocols without using random oracles. The other interesting research question is to use Timed-Ephemerizer as a tool to solve practical security problems. Note that, as an application of Ephemerizer, Perlman [11] proposes a file system that supports high availability of data with assured delete.

References

1. M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. K. Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289. Springer, 2004.
2. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
3. J. Cathalo, B. Libert, and J.-J. Quisquater. Efficient and non-interactive timed-release encryption. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *Proceedings of the 7th International Conference on Information and Communications Security*, volume 3783 of *Lecture Notes in Computer Science*, pages 291–303. Springer-Verlag, 2005.
4. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO 1991*, volume 576 of *LNCS*, pages 445–456. Springer, 1991.
5. A. W. Dent and Q. Tang. Revisiting the security model for timed-release encryption with pre-open capability. In J. A. Garay, A. K. Lenstra, M. Mambo, and R. Peralta, editors, *Information Security, 10th International Conference, ISC 2007*, volume 4779 of *Lecture Notes in Computer Science*, pages 158–174. Springer, 2007.
6. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest We Remember: Cold Boot Attacks on Encryption Keys. In P. C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.
7. Y. Hwang, D. Yum, and P. Lee. Timed-release encryption with pre-open capability and its application to certified e-mail system. In J. Zhou, J. Lopez, R. Deng, and F. Bao, editors, *Proceedings of the 8th International Information Security Conference (ISC 2005)*, volume 3650 of *Lecture Notes in Computer Science*, pages 344–358. Springer, 2005.
8. T. C. May. *Time-release crypto*, 1993.
9. S. K. Nair, M. T. Dashti, B. Crispo, and A. S. Tanenbaum. A Hybrid PKI-IBC Based Ephemerizer System. In H. S. Venter, M. M. Eloff, L. Labuschagne, J. H. P. Eloff, and R. von Solms, editors, *New Approaches for Security, Privacy and Trust in Complex Environments, Proceedings of the IFIP TC-11 22nd International Information Security Conference (SEC 2007)*, volume 232 of *IFIP*, pages 241–252. Springer, 2007.
10. Department of Defense of the United States. *National Industrial Security Program Operating Manual (NISPOM)*, 2006. DoD 5220.22-M.
11. R. Perlman. File system design with assured delete. In *SISW '05: Proceedings of the Third IEEE International Security in Storage Workshop*, pages 83–88. IEEE Computer Society, 2005.
12. R. Perlman. The Ephemerizer: Making Data Disappear. *Journal of Information System Security*, 1(1):51–68, 2005.
13. R. Perlman. The Ephemerizer: Making Data Disappear. Technical Report TR-2005-140, Sun Microsystems, Inc., 2005.
14. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report Tech. Report MIT/LCS/TR-684, MIT LCS, 1996.
15. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology, Proceedings of CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1985.
16. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006.

Appendix A: Proofs for the Lemmas

Proof sketch of Lemma 1. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the attack game depicted in Figure 1.

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from $\mathcal{A}$. We assume the challenger simulates the hash function H_1 as follows. The challenger maintains a list of vectors, each of them containing a request message, an element of $\mathbb{G}$ (the hash-code for this message), and an element of the form $ID_T\|t$. After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\mathbb{G}$; otherwise, the challenger returns g^y, where y a randomly chosen element of $\mathbb{Z}_p$, and stores the new vector in the list. Other hash functions are simulated in a similar way.

On receiving a $\mathsf{TimeExt}$ oracle query with the input t, the challenger answers PK_T^y given that $\mathsf{H}_1(ID_T\|t) = g^y$. Let $\delta_0 = \Pr[b' = b]$, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

Game_1: In this game, the challenger performs in the same way as in Game_0 except for the generation of the challenge C_b.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^* \in_R \mathcal{X},\ R \in \mathbb{G}_1,\ C_1^* = g^{r_1^*},\ C_2^* = g^{r_2^*},\ C_3^* = \mathsf{H}_4(C_1^*\|C_2^*)^{r_1^*},$$

$$C_4^* = M_b \oplus \mathsf{H}_3(R),\ C_5^* = \mathsf{Encrypt}_2(X^*, PK_U),\ C_6^* = \mathsf{H}_5(X^*) \oplus (C_1^*\|C_2^*\|C_3^*\|C_4^*),$$

$$C_7^* = \mathsf{H}_6(X^*\|C_1^*\|C_2^*\|C_3^*\|C_4^*\|C_5^*\|C_6^*),\ C_b = (C_5^*, C_6^*, C_7^*).$$

Let δ_1 be the probability that the challenger successfully ends and $b' = b$ in Game_1. As $R \in_R \mathbb{G}_1$ and H_3 is modeled as a random oracle, the equation $|\delta_1 - \frac{1}{2}| = 0$ holds.

With respect to the generation of C_b, from Game_0 to Game_1, the only modification is that $\hat{e}(\mathsf{H}_2(ID_E\|t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T\|t_{int}^*), C_2^*)^{SK_T}$ has been replaced with R, where $R \in_R \mathbb{G}_1$. As a result, Game_1 is identical to Game_0 unless $\hat{e}(\mathsf{H}_2(ID_E\|t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T\|t_{int}^*), C_2^*)^{SK_T}$ has been queried to H_3. Note that $SK_E^{(0)}$ is not required in answering the $\mathsf{TimeExt}$ oracle queries. We immediately obtain $|\delta_1 - \delta_0| = \epsilon'$ where ϵ' is negligible based on the BDH assumption. The lemma now follows. □

Proof sketch of Lemma 2. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the attack game depicted in Figure 1. The security proof is done through a sequence of games [16].

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from $\mathcal{A}$. Let $\delta_0 = \Pr[b' = b]$, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

Before moving ahead, we first describe the following claims. The verifications of these two claims can be done straightforwardly in the random oracle model given the encryption schemes $\mathcal{E}_1$ and $\mathcal{E}_2$ are one-way permutation.

Claim. Before the adversary queries SK_U, given a D-type $\mathsf{Retrieve}$ query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_i})$, given that any of $C_5 = C_5^*$, $C_6 = C_6^*$, or $C_7 = C_7^*$ holds, then the challenger will accept the request with only a negligible probability if one of them does not hold.

Claim. Given an E-type $\mathsf{Retrieve}$ query with the input $(C' = (C_5', C_6', C_7'), TS_{t_{int}}, t_{eph_i})$[2], given that any of C_5', C_6', or C_7' is equal to the corresponding element from the output of a D-type $\mathsf{Retrieve}$ query, then the challenger will accept the request with only a negligible probability if any of them is not equal to the corresponding element of the same output.

Claim. Given an E-type $\mathsf{Retrieve}$ query with the input $(C' = (C_5', C_6', C_7'), TS_{t_{int}}, t_{eph_i})$, given that C' is not the output of a D-type $\mathsf{Retrieve}$ query, then the the probability $C_1' = C_1^*$ is negligible.

Game_1: In this game, the challenger performs in the same way as in Game_0 except for the following event Evn: For any E-type $\mathsf{Retrieve}$ query with the input $(C' = (C_5', C_6', C_7'), t_{eph_i})$, the challenger rejects the request if $C_1' = C_1^*$, where $Y = \mathsf{Decrypt}_1(C_5', SK_{t_{eph_i}}^{(1)})$ and $C_1' \| C_2' \| C_3' \| C_4' = \mathsf{H}_9(Y) \oplus C_6'$, and C' is not one of the output of D-type $\mathsf{Retrieve}$ queries.

Let δ_1 be the probability that the challenger successfully ends and $b' = b$ in Game_1. From the third claim we made in Game_0, we have $|\delta_1 - \delta_0| = \epsilon_1$ is negligible.

Game_2: In this game, the challenger performs in the same way as in Game_1 except for the following. For any E-type $\mathsf{Retrieve}$ query with the input $(C' = (C_5', C_6', C_7'), TS_{t_{int}}, t_{eph_i})$, the challenger returns $T \in \{0,1\}^n$ if $C_1' = C_1^*$ where $Y = \mathsf{Decrypt}_1(C_5', SK_{t_{eph_i}}')$ and $C_1' \| C_2' \| C_3' \| C_4' = \mathsf{H}_9(Y) \oplus C_6'$.

The game Game_2 is identical to Game_1 unless the following event Evn occurs: For some aforementioned E-type $\mathsf{Retrieve}$ oracle query with the input $(C' = (C_5', C_6', C_7'), TS_{t_{int}}, t_{eph_i})$, the adversary has queried H_8 with the input $Y \| C_1' \| C_2' \| C_3' \| C_4'$. As the encryption scheme $\mathcal{E}_2$ is one-way permutation and the hash functions are random oracles, the probability $\Pr[Evn]$ is negligible. Let δ_2 be the probability that the challenger successfully ends and $b' = b$ in Game_2. Therefore, we have $|\delta_2 - \delta_1| \leq \epsilon_2 = \Pr[Evn]$ is negligible.

Game_3: In this game, the challenger performs in the same way as in Game_2 except for answering the $\mathsf{Retrieve}$ oracle queries.

1. Given an E-type $\mathsf{Retrieve}$ query with the input $(C' = (C_5', C_6', C_7'), TS_{t_{int}}, t_{eph_i})$, where C' is not the output of a D-type $\mathsf{Retrieve}$ query, the challenger first

[2] Note that in the formal definition of the semantic security against Type-II adversary, the input to an E-type $\mathsf{Retrieve}$ query is t_{eph_i}. However, as $\mathsf{Retrieve}$ is an interactive algorithm and the adversary is assumed to control the communication channels, hence, we let the adversary to choose other inputs. Specifically, in our protocol, the adversary can choose $(C' = (C_5', C_6', C_7')$ and $t_{eph_i})$.

checks whether or not there is an query

$$\tilde{Y}||\tilde{C'_1}||\tilde{C'_2}||\tilde{C'_3}||\tilde{C'_4}||\tilde{C'_5}||\tilde{C'_6}$$

to the oracle H_7 such that $C'_5 = \mathsf{Encrypt}_1(\tilde{Y}, PK^{(2)}_{t_{eph_i}})$,

$$C'_6 = \tilde{C'_6},\ \mathsf{H}_9(\tilde{Y}) \oplus C'_6 = \tilde{C'_1}||\tilde{C'_2}||\tilde{C'_3}||\tilde{C'_4},\ \text{and}\ \ C'_7 = \mathsf{H}_7(\tilde{Y}||\tilde{C'_1}||\tilde{C'_2}||\tilde{C'_3}||\tilde{C'_4}||\tilde{C'_5}||\tilde{C'_6}).$$

If the input exists, the challenger proceeds. If $\hat{e}(\tilde{C'_3}, g) \neq \hat{e}(\tilde{C'_1}, \mathsf{H}_4(\tilde{C'_1}||\tilde{C'_2}))$, it aborts; otherwise it returns C'', where

$$C'' = \mathsf{H}_8(\tilde{Y}||\tilde{C'_1}||\tilde{C'_2}||\tilde{C'_3}||\tilde{C'_4}) \oplus \tilde{C'_3} \oplus \mathsf{H}_3(\hat{e}(\tilde{C'_1}, \mathsf{H}_2(PK_{t_{eph_i}}))^{\tilde{r'_1}} \cdot \hat{e}(TS_{t_{int}}, \tilde{C'_2})).$$

Note that the challenger retrieves $\tilde{r'_1}$ such that $g^{\tilde{r'_1}} = \tilde{C'_1}$.

Let δ_3 be the probability that the challenger successfully ends and $b' = b$ in Game_3. The game Game_3 is identical to Game_2 unless the following event Evn occurs in answering the E-type $\mathsf{Retrieve}$ oracle queries: For any query with some input $(C' = (C'_5, C'_6, C'_7), t_{eph_i})$: for $\tilde{Y} = \mathsf{Decrypt}(C'_5, SK^{(1)}_{t_{eph_i}})$, an oracle query to H_7 with the input $\tilde{Y}||\tilde{C'_1}||\tilde{C'_2}||\tilde{C'_3}||\tilde{C'_4}||\tilde{C'_5}||\tilde{C'_6}$ returns C'_7. As H_7 is modeled as a random oracle, the probability $\Pr[Evn]$ is negligible. Let δ_3 be the probability that the challenger successfully ends and $b' = b$ in Game_3. Therefore, we have $|\delta_3 - \delta_2| \leq \epsilon_3 = \Pr[Evn]$ is negligible.

Game_4: In this game, the challenger performs in the same way as in Game_3 except that the challenge C_b is computed as follows.

$$r^*_1, r^*_2 \in_R \mathbb{Z}_p,\ X^* \in_R \mathcal{X},\ R \in \mathbb{G}_1,\ C^*_1 = g^{r^*_1},\ C^*_2 = g^{r^*_2},\ C^*_3 = \mathsf{H}_4(C^*_1||C^*_2)^{r^*_1},$$

$$C^*_4 = M_b \oplus \mathsf{H}_3(R),\ C^*_5 = \mathsf{Encrypt}_2(X^*, PK_U),\ C^*_6 = \mathsf{H}_5(X^*) \oplus (C^*_1||C^*_2||C^*_3||C^*_4),$$

$$C^*_7 = \mathsf{H}_6(X^*||C^*_1||C^*_2||C^*_3||C^*_4||C^*_5||C^*_6),\ C_b = (C^*_5, C^*_6, C^*_7).$$

Let δ_4 be the probability that the challenger successfully ends and $b' = b$ in Game_4. As $R \in_R \mathbb{G}_1$, the equation $|\delta_4 - \frac{1}{2}| = 0$ holds.

With respect to the generation of C_b, from Game_3 to Game_4, the only modification is that $\hat{e}(\mathsf{H}_2(ID_E||t_{eph_j}), C^*_1)^{SK^{(0)}_E} \cdot \hat{e}(\mathsf{H}_1(ID_T||t^*_{int}), C^*_2)^{SK_T}$ has been replaced with R, where $R \in_R \mathbb{G}_1$. As a result, Game_4 is identical to Game_3 unless $\hat{e}(\mathsf{H}_2(ID_E||t_{eph_j}), C^*_1)^{SK^{(0)}_E} \cdot \hat{e}(\mathsf{H}_1(ID_T||t^*_{int}), C^*_2)^{SK_T}$ has been queried to H_3. We immediately obtain $|\delta_4 - \delta_3| \leq \epsilon_4$ which is negligible based on the BDH assumption.

In summary, we have $|\delta_0 - \delta_4| = \epsilon \leq \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4$. which are negligible. As a result, ϵ is negligible, and the lemma now follows. □

Proof sketch of Lemma 3. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the attack game depicted in Figure 3.

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from $\mathcal{A}$. Note that the challenge C_b is computed as follows. Let $\delta_0 = \Pr[b' = b]$, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

Game_1: In this game, the challenger performs in the same way as in Game_0 except for the following. Given a D-type $\mathsf{Retrieve}$ query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$, the challenger answers as the following.

1. If $C = C_b$, the challenger returns C', where

$$M' \in_R \{0,1\}^n,\ C'_1 = C^*_1,\ C'_2 = C^*_2,\ C'_3 = C^*_3,\ C'_4 = M' \oplus C^*_4,$$
$$Y \in_R \mathcal{Y},\ C'_5 = \mathsf{Encrypt}_1(Y, PK^{(1)}_{t_{eph_j}}),\ C'_6 = \mathsf{H}_9(Y) \oplus (C'_1||C'_2||C'_3||C'_4),$$
$$C'_7 = \mathsf{H}_7(Y||C'_1||C'_2||C'_3||C'_4||C'_5||C'_6),\ C' = (C'_5, C'_6, C'_7).$$

2. Otherwise, the challenger first checks whether or not there is a query with the input
$$\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6$$
to the oracle H_5 such that $C_5 = \mathsf{Encrypt}_2(\tilde{X}, PK_U)$,
$$C_6 = \tilde{C}_6,\ \mathsf{H}_5(\tilde{X}) \oplus C_6 = \tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4, \text{ and } C_7 = \mathsf{H}_6(\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6).$$
If the input exists, the challenger returns C', where
$$M' \in_R \{0,1\}^n,\ C'_1 = \tilde{C}_1,\ C'_2 = \tilde{C}_2,\ C'_3 = \tilde{C}_3,\ C'_4 = M' \oplus \tilde{C}_4,$$
$$C'_5 = \mathsf{Encrypt}_1(Y, PK^{(1)}_{t_{eph_j}}),\ C'_6 = \mathsf{H}_9(Y) \oplus (C'_1||C'_2||C'_3||C'_4),$$
$$C'_7 = \mathsf{H}_7(Y||C'_1||C'_2||C'_3||C'_4||C'_5||C'_6),\ C' = (C'_5, C'_6, C'_7).$$
Otherwise, the challenger rejects the quest.

The game Game_1 is identical to Game_0 unless the following event Evn occurs in answering the D-type $\mathsf{Retrieve}$ oracle queries: For any query with some input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$: for $\tilde{X} = \mathsf{Decrypt}_2(C_5, SK_U)$, an oracle query to H_6 with the input $\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6$ returns C_7. As H_6 is modeled as a random oracle, the probability $\Pr[Evn]$ is negligible. Let δ_1 be the probability that the challenger successfully ends and $b' = b$ in Game_1. Therefore, we have $|\delta_1 - \delta_0| \leq \epsilon_1 = \Pr[Evn]$ is negligible.

Game_2: In this game, the challenger performs in the same way as in Game_1 except that the challenge C_b is computed as follows.

$$r^*_1, r^*_2 \in_R \mathbb{Z}_p,\ X^*, X^\dagger \in_R \mathcal{X},\ C^*_1 = g^{r^*_1},\ C^*_2 = g^{r^*_2},\ C^*_3 = \mathsf{H}_4(C^*_1||C^*_2)^{r^*_1},$$

$$\begin{aligned} C^*_4 &= M_b \oplus \mathsf{H}_3(\hat{e}(\mathsf{H}_2(PK^{(0)}_{t_{eph_i}}), PK^{(0)}_E)^{r^*_1} \cdot \hat{e}(\mathsf{H}_1(ID_T||t^*_{int}), PK_T)^{r^*_2}) \\ &= M_b \oplus \mathsf{H}_3(\hat{e}(\mathsf{H}_2(ID_E||t_{eph_i}), C^*_1)^{SK^{(0)}_E} \cdot \hat{e}(\mathsf{H}_1(ID_T||t^*_{int}), C^*_2)^{SK_T}), \end{aligned}$$

$$C_5^* = \mathsf{Encrypt}_2(X^\dagger, PK_U),\ C_6^* = \mathsf{H}_5(X^*) \oplus (C_1^* \| C_2^* \| C_3^* \| C_4^*),$$
$$C_7^* = \mathsf{H}_6(X^* \| C_1^* \| C_2^* \| C_3^* \| C_4^* \| C_5^* \| C_6^*),\ C_b = (C_5^*, C_6^*, C_7^*).$$

The game Game_2 is identical to Game_1 unless the following event Evn occurs: the adversary queries H_5 with $0\|X^\dagger$ or H_6 with $0\|X^\dagger\| * \| * \| * \| * \| * \|*$. As $\mathcal{E}_2$ is one-way and $\mathsf{H}_5, \mathsf{H}_6$ are random oracles, the probability $\Pr[Evn]$ is negligible. Let δ_2 be the probability that the challenger successfully ends and $b' = b$ in Game_2. Therefore, we have $|\delta_2 - \delta_1| \leq \epsilon_2 = \Pr[Evn]$ is negligible.

Game_3: In this game, the challenger performs in the same way as in Game_3 except that the challenge C_b is computed as follows.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^*, X^\dagger \in_R \mathcal{X},\ R \in \mathbb{G}_1,\ C_1^* = g^{r_1^*},\ C_2^* = g^{r_2^*},\ C_3^* = \mathsf{H}_4(C_1^* \| C_2^*)^{r_1^*},$$

$$C_4^* = M_b \oplus \mathsf{H}_3(R),\ C_5^* = \mathsf{Encrypt}_2(X^\dagger, PK_U),\ C_6^* = \mathsf{H}_5(X^*) \oplus (C_1^* \| C_2^* \| C_3^* \| C_4^*),$$
$$C_7^* = \mathsf{H}_6(X^* \| C_1^* \| C_2^* \| C_3^* \| C_4^* \| C_5^* \| C_6^*),\ C_b = (C_5^*, C_6^*, C_7^*).$$

Regardless of the change, the game Game_3 is identical to Game_2 as $X^* \in_R \mathcal{X}$ and $\mathsf{H}_5, \mathsf{H}_6$ are random oracles. Let δ_3 be the probability that the challenger successfully ends and $b' = b$ in Game_3. As $R \in_R \mathbb{G}_1$, the equation $|\delta_2 - \frac{1}{2}| = |\delta_3 - \frac{1}{2}| = 0$ holds.

In summary, we have $|\delta_0 - \delta_3| = \epsilon \leq \epsilon_1 + \epsilon_2$. which are negligible. As a result, ϵ is negligible, and the lemma now follows. □

Appendix B: Review of one Hybrid PKI-IBC Protocol

The hybrid PKI-IBC protocol [9] involves the following types of entities: data generator, data consumer, and Ephemerizer. The algorithms are defined as follows.

- $\mathsf{Setup}_E(\ell)$: The Ephemerizer generates a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, a generator $P \in_R \mathbb{G}_1$, a long-term private key $SK_E \in_R \mathbb{Z}_p$ and the public key $PK_E = SK_E P$, two hash functions

  $$\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1,\ \ \mathsf{H}_2 : \mathbb{G}_2 \to \{0,1\}^n,$$

  and a set of ephemeral tuples $(KeyID_{t_{eph_j}}, PK_{t_{eph_j}}, SK_{t_{eph_j}}, t_{exp_j})$ where $KeyID_{t_{eph_j}}$ is the identifier of this tuple, t_{eph_j} is the expiration time, and $PK_{t_{eph_j}} = SK_{t_{eph_j}} P$. Suppose that $\mathbb{G}_1$ is additive and $\mathbb{G}_T$ is multiplicative. Suppose also that the Ephemerizer possesses the identity ID_E.
- $\mathsf{Setup}_U(\ell)$: The data consumer generates a key pair (PK_U, SK_U) for a public key encryption scheme $\mathcal{E}_2$ with the encryption/decryption algorithms $(\mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$. The data consumer also selects a symmetric key encryption scheme

  $$\mathcal{E}_1 = (\mathsf{Encrypt}_1, \mathsf{Decrypt}_1),$$

  which will be used to encrypt data in the system.

– $\mathsf{Generate}(M, PK_U, PK_{t_{eph_j}})$: The data generator sends $(KeyID_{t_{eph_j}}, C)$ to the data consumer, where

$$r \in_R \mathbb{Z}_p,\ C_m = \mathsf{Encrypt}_1(M, K),\ C_k = \mathsf{Encrypt}_2(K, PK_U),$$

$$ID_{t_{eph_j}} = ID_E \| Expiry : t'_{exp_j},\ Q_{t_{eph_j}} = \hat{e}(\mathsf{H}_1(ID_{t_{eph_j}}), PK_{t_{eph_j}}),$$

$$C_{t_{eph_j}} = (rP, C_k \oplus \mathsf{H}_2((Q_{t_{eph_j}})^r)), \tag{1}$$

$$C^{\dagger}_{t_{eph_j}} = \mathsf{Encrypt}_2(ID_{t_{eph_j}} \| C_{t_{eph_j}}, PK_U),\ C = (C_m, C^{\dagger}_{t_{eph_j}}). \tag{2}$$

It is required t'_{exp_j} should be smaller than t_{exp_j} which is the expiration time of $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$.

– $\mathsf{Retrieve}(C, SK_U; SK_{t_{eph_j}}, SK_E)$:

1. The data consumer first decrypts $C^{\dagger}_{t_{eph_j}}$ to obtain $ID_{t_{eph_j}}$ and $C_{t_{eph_j}}$, and then computes and sends $(KeyID_{t_{eph_j}}, ID'_{t_{eph_j}}, C'_{t_{eph_j}})$ to the Ephemerizer, where

$$ID'_{t_{eph_j}} \in_R \{0,1\}^*,\ Q'_{t_{eph_j}} = \hat{e}(\mathsf{H}_1(ID'_{t_{eph_j}}), PK_E),$$

$$r \in_R \mathbb{Z}_p,\ C'_{t_{eph_j}} = (r'P, (ID_{t_{eph_j}} \| K') \oplus \mathsf{H}_2((Q'_{t_{eph_j}})^{r'})). \tag{3}$$

2. If the ephemeral key $SK_{t_{eph_j}}$ associated with $KeyID_{t_{eph_j}}$ has not expired, the Ephemerizer decrypts $C'_{t_{eph_j}}$ to obtain $ID_{t_{eph_j}}$ and K' as follows

$$\begin{aligned} ID_{t_{eph_j}} \| K' = (ID_{t_{eph_j}} \| K') \oplus \mathsf{H}_2((Q'_{t_{eph_j}})^{r'}) \oplus \\ \mathsf{H}_2(\hat{e}(\mathsf{H}_1(ID'_{t_{eph_j}}), r'P)^{SK_E}). \end{aligned} \tag{4}$$

   It then computes and sends $C''_{t_{eph_j}}$ to the data consumer, where

$$C''_{t_{eph_j}} = \mathsf{Encrypt}_1(SK_{t_{eph_j}} \mathsf{H}_1(ID_{t_{eph_j}}), K'). \tag{5}$$

3. The data consumer decrypts $C''_{t_{eph_j}}$ to obtain $SK_{t_{eph_j}} \mathsf{H}_1(ID_{t_{eph_j}})$, and then decrypts $C_{t_{eph_j}}$ to obtain C_k as follows

$$C_k = C_k \oplus \mathsf{H}_2((Q_{t_{eph_j}})^r) \oplus \mathsf{H}_2(\hat{e}(rP, SK_{t_{eph_j}} \mathsf{H}_1(ID_{t_{eph_j}}))). \tag{6}$$

   The data consumer then sequentially decrypts C_k and C_m to obtain M as follows:

$$K = \mathsf{Decrypt}_2(C_k, SK_U),\ \ M = \mathsf{Decrypt}_1(C_m, K). \tag{7}$$

In [9], no rigorous analysis has been done for this protocol. As to the security, we have the following observations.

- When the data generator constructs $ID_{t_{eph_j}} = ID_E || Expiry : t'_{exp_j}$, she chooses an ephemeral public key $PK_{t_{eph_j}}$ where $t'_{exp_j} < t_{exp_j}$. This means that at the time between t'_{exp_j} and t_{exp_j}, if an adversary compromises both the Ephemerizer and the data consumer, then he is able to recover M. This observation implies that the expiration time for the ciphertext C is in fact t_{exp_j} instead of t'_{exp_j}.
- There is an attack scenario, in which an adversary can recover any message even after the expiration time.
  1. At one point, the adversary compromises the Ephemerizer and obtains the long-term private key SK_E.
  2. From Equations (3) and (4), by decrypting $C'_{t_{eph_j}}$ the adversary could obtain any $ID_{t_{eph_j}} || K'$ (regardless of the expiration time t_{eph_j}). From Equation (5), the adversary can recover $SK_{t_{eph_j}} \mathsf{H}_1(ID_{t_{eph_j}})$ by decrypting $C''_{t_{eph_j}}$ using K'.
  3. For any $(KeyID_{t_{eph_j}}, C)$, if the adversary can compromise the data consumer then he is able to recover M from Equations (1),(2),(6),(7). Note that the decryption does not need the involvement of any ephemeral keys from the Ephemerizer.
- Note the fact there is no authentication between the Ephemerizer and the data consumer. There is another attack scenario, in which an adversary can recover any message (even after the expiration time) as long as he can compromise the data consumer.
  1. At one point, the adversary impersonates an honest data consumer to run the Retrieve algorithm with the Ephemerizer. In more detail, by sending $(KeyID_{t_{eph_j}}, ID'_{t_{eph_j}}, C'_{t_{eph_j}})$ (generated by himself) to the Ephemerizer, the adversary can obtain $SK_{t_{eph_j}} \mathsf{H}_1(ID_{t_{eph_j}})$ trivially before the expiration time t_{eph_j}. It is worth stressing that, to do this, the adversary does not need to compromise any party in the system.
  2. Later on, for any $(KeyID_{t_{eph_j}}, C)$, if the adversary can compromise the data consumer and obtain SK_U, then he is able to recover M. The computation is straightforward from Equations (1),(2),(6),(7). Clearly, the decryption does not need the involvement of any ephemeral keys from the Ephemerizer.