From Ephemerizer to Timed-Ephemerizer: Achieve Assured Lifecycle Enforcement for Sensitive Data

Qiang Tang

APSIA group, SnT, University of Luxembourg. 6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
Email: qiang.tang@uni.lu

The concept of Ephemerizer, proposed by Perlman, is a cryptographic primitive for assured data deletion. With an Ephemerizer protocol, data in persistent storage devices will always be encrypted simultaneously using an ephemeral public key of the Ephemerizer (an entity which will publish a set of ephemeral public keys and periodically delete the expired ones) and the long-term public key of a user. An Ephemerizer protocol enables the user to securely decrypt the encrypted data without leaking any information to the Ephemerizer. So far, no security model has ever been proposed for this primitive and existing protocols have not been studied formally. Not surprisingly, we show that some existing Ephemerizer protocols possess security vulnerabilities. In this paper, we review the notion of Timed-Ephemerizer, which can be regarded as a hybrid primitive by combining Ephemerizer and Timed-Release Encryption. Compared with an Ephemerizer protocol, a Timed-Ephemerizer protocol further guarantees that data will only be released after a pre-defined disclosure time. Moreover, we revisit a security model for Timed-Ephemerizer and adapt it for Ephemerizer. We also revise a previous Timed-Ephemerizer protocol by Tang and prove its security in the security model.

Keywords: Ephemerizer; storage privacy; assured lifecycle

Received x; revised y

1. INTRODUCTION

Rapid growth of information technology has greatly facilitated individuals and enterprizes to generate and store sensitive data (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 backed up 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 [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 [2].
2. Backups of encrypted sensitive data and cryptographic keys often reside in many devices. Consequently, it is difficult to guarantee 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. To reduce the potential risks facing sensitive data, it is crucial to define an expiration time and strictly enforce secure deletion afterwards. Subsequently, an effective cryptographic protocol is needed for the enforcement.

Ephemerizer, proposed by Perlman [3, 4] and further studied by Nair *et al.* [5], has shown a promising direction towards a practical solution to the above problem. At the core of Ephemerizer is a key management service provided by an entity, referred to as the Ephemerizer, which will publish a set of ephemeral public keys and securely delete the expired ones periodically. With an Ephemerizer protocol, data in persistent storage devices will always be encrypted simultaneously using an ephemeral public key of the Ephemerizer and the long-term public key of a user. The novelty of a *secure* Ephemerizer protocol is that it enables the user to securely decrypt the encrypted data without leaking any information to the Ephemerizer. If we assume that the plaintext data only reside in volatile storage devices such as memory, then an Ephemerizer protocol guarantees that expired data will remain unrecoverable even if the user's persistent storage, the user's long-term private key, and the unexpired private keys of the Ephemerizer have been compromised.

1.1. Problem Statement

So far, no security model has ever been proposed for Ephemerizer and existing protocols have not been analyzed formally. As a result, even if an existing Ephemerizer protocol is employed, it is not clear what kind of security guarantee will be provided. To gain confidence in these protocols, we need to propose a formal security model and conduct corresponding security analysis. The other concern is that an Ephemerizer protocol is only supposed to provide assured deletion but not assured initial disclosure, which however could be a very useful feature in some practical applications. In a security guideline published by the Cloud Security Alliance[1], thirteen domains of interest have been identified for applications in the cloud computing environment, one of which is *information lifecycle management*. As such, protocols providing assured lifecycle (marked by an assured initial disclosure and an assured deletion) will be more appealing than those providing only assured deletion. We illustrate this by an outsourcing data security example in Section 6.2.

It is worth noting that Ephemerizer and its variants only guarantee the lifecycle of data release through encryption. Once data is decrypted (e.g. residing in clear in memory), other complementary solutions are needed to securely delete the data (e.g. the state re-incarnation technique [6]). We do not consider such techniques in this work.

1.2. Our Contribution

This is an extended version of the conference paper [7], in which the author only introduced the concept of Timed-Ephemerizer and gave an instantiation. Our extentions fall into the following aspects.

Firstly, we show that some Ephemerizer protocols in [5, 3, 4] possess security vulnerabilities. Specifically, we show that the Ephemerizer protocol using blind decryption technique in [3, 4] is vulnerable to attacks from a curious Ephemerizer. More seriously, we show that the hybrid PKI-IBC Ephemerizer protocol in [5] does not achieve assured deletion, i.e. an adversary can recover expired data.

Secondly, we provide a general overview of Timed-Ephemerizer. The workflow is illustrated in Fig. 1. Data in persistent storage devices will always be encrypted using an ephemeral public key of the Ephemerizer, the public key of the user, and the long-term public key of the time server (which will publish timestamps periodically). As a security guarantee, a Timed-Ephemerizer protocol enables the user to securely decrypt the encrypted data only during a pre-defined time slot, namely before the expiration of the ephemeral private key and the after the release of the timestamp from the time server, yet without leaking any information to the Ephemerizer or the time server. If the time server's private key is made public (or, the assured initial disclosure property is disabled), then Timed-Ephemerizer becomes Ephemerizer. Based on the security model for Timed-Ephemerizer, we present a security model for Ephemerizer.

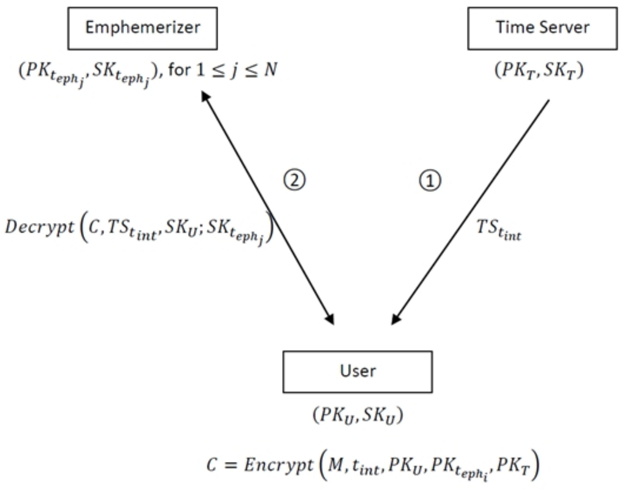
Emphemerizer
$(PK_{teph_j}, SK_{teph_j})$, for $1 \le j \le N$
Time Server
(PK_T, SK_T)
②
①
$Decrypt\left(C, TS_{t_{int}}, SK_U; SK_{teph_j}\right)$
$TS_{t_{int}}$
User
(PK_U, SK_U)
$C = Encrypt\left(M, t_{int}, PK_U, PK_{teph_i}, PK_T\right)$

FIGURE 1. An Illustration of Timed-Ephemerizer

Thirdly, we revise the Timed-Ephemerizer protocol [7] with hash function usage and prove its security in the proposed security model.

Fourthly, we compare Timed-Ephemerizer with other solutions such as Vanish [8]. As an application, we show that Timed-Ephemerizer is a useful tool for users to enforce information lifecycle management in outsourcing activities.

1.3. Organization

The rest of the paper is organized as follows. In Section 2 we briefly present some notations and

[1]http://www.cloudsecurityalliance.org/

basics about pairing. In Section 3 we show that some existing Ephemerizer protocols possess security vulnerabilities. In Section 4 we review the concept of Timed-Ephemerizer and its security properties. In Section 5 we revise the Timed-Ephemerizer protocol [7] and prove its security. In Section 6, we present some further remarks on Timed-Ephemerizer. In Section 7 we briefly review the relevant works on Ephemerizer and Timed-Release Encryption. In Section 8 we conclude the paper.

2. PRELIMINARY

Throughout the paper, we use ℓ to denote the security parameter. The notation $u \in_R S$ means u is chosen from the set S uniformly at random. 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.

We now review some basics about pairing and the related assumptions. More detailed information can be found in the seminal paper [9]. 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 2.1. *We say that the (decision) BDH assumption holds in $\mathbb{G}$ if no polynomial-time algorithm has a non-negligible advantage in solving the (decision) BDH problem.*

Besides these computational/decisional assumptions, the Knowledge of Exponent (KE) assumption is also used in a number of papers (e.g. [10, 11]). The KE assumption is defined as follows.

Definition 2.2. *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 $g^c = C$.*

3. REVIEW OF EXISTING EPHEMERIZER PROTOCOLS

An Ephemerizer protocol involves two types of entities: users and an Ephemerizer. The idea behind of this primitive is quite simple. The Ephemerizer chooses a set of time timestamps, say t_{eph_j} $(1 \leq j \leq \mathcal{N})$ where $\mathcal{N}$ is an integer, and correspondingly generates a set of ephemeral key pairs $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$. For $1 \leq j, k \leq \mathcal{N}$, it is assumed that if $j < k$ then $t_{eph_j} < t_{eph_k}$ (i.e. t_{eph_k} is later than t_{eph_j}). Data is encrypted using a symmetric key, which will be double-encrypted using one ephemeral public key of the Ephemerizer and the public key of the user. The decryption is an interactive algorithm between the user and the Emphemerizer. It is required that, at time t_{eph_j}, the Ephemerizer will delete $SK_{t_{eph_j}}$, so that any ciphertext generated under the ephemeral public key $PK_{t_{eph_j}}$ will become unrecoverable afterwards.

In this section, we point out some vulnerabilities of the Ephemerizer protocol which uses the blind decryption technique in [3, 4] and the hybrid PKI-IBC Ephemerizer protocol in [5].

3.1. Ephemerizer Protocol using Blind Decryption

3.1.1. *Description of the Protocol*

The Ephemerizer protocol using blind decryption [3, 4] consists of algorithms $(\mathsf{Setup}_E, \mathsf{Setup}_U, \mathsf{Encrypt}, \mathsf{Decrypt})$, which are defined as follows.

- $\mathsf{Setup}_E(\ell)$: The Ephemerizer generates a set of tuples
$$(KeyID_{t_{eph_j}}, PK_{t_{eph_j}}, SK_{t_{eph_j}}, t_{eph_j}),$$
where $1 \leq j \leq \mathcal{N}$, $KeyID_{t_{eph_j}}$ is the identifier of this tuple, $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ is a key pair of a public key encryption scheme $\mathcal{E}_1$ with the algorithms $(\mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$, and t_{eph_j} is the expiration time of the key pair.
- $\mathsf{Setup}_U(\ell)$: A user generates a key pair (PK_U, SK_U) for a public key encryption scheme $\mathcal{E}_2$ with the algorithms $(\mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$. The user also selects a symmetric key encryption scheme
$$\mathcal{E}_0 = (\mathsf{Encrypt}_0, \mathsf{Decrypt}_0),$$

which has the key space $\mathcal{K}$ and will be used to encrypt data in the system.

- $\mathsf{Encrypt}(M, PK_U, PK_{t_{eph_j}})$: The ciphertext is $(KeyID_{t_{eph_j}}, C, PK_{t_{eph_j}})$, where $K \in_R \mathcal{K}$,

$$C_m = \mathsf{Encrypt}_0(M, K),\ C_k = \mathsf{Encrypt}_1(K, PK_{t_{eph_j}}),$$

$$C_{t_{eph_j}} = \mathsf{Encrypt}_2(C_k, PK_U),\ C = (C_m, C_{t_{eph_j}}).$$

- $\mathsf{Decrypt}(C, SK_U; SK_{t_{eph_j}})$:
  1. The user generates an ephemeral function pair $(\mathsf{Blind}, \mathsf{Unblind})$ satisfying the following homomorphic property:
  $$K = \mathsf{Unblind}(\mathsf{Decrypt}_1(\mathsf{Blind}(C_k), SK_{t_{eph_j}})).$$
  2. The user then decrypts $C_{t_{eph_j}}$ to obtain C_k, and then computes and sends $(KeyID_{t_{eph_j}}, C'_{t_{eph_j}})$ to the Ephemerizer, where
  $$C'_{t_{eph_j}} = \mathsf{Blind}(C_k).$$
  3. If the ephemeral key $SK_{t_{eph_j}}$ associated with $KeyID_{t_{eph_j}}$ has not expired, the Ephemerizer decrypts $C'_{t_{eph_j}}$ and sends $C''_{t_{eph_j}}$ to the user, where
  $$\begin{aligned} C''_{t_{eph_j}} &= \mathsf{Decrypt}_1(C'_{t_{eph_j}}, SK_{t_{eph_j}}) \\ &= \mathsf{Decrypt}_1(\mathsf{Blind}(C_k), SK_{t_{eph_j}}). \end{aligned}$$
  4. The user obtains M as follows
  $$K = \mathsf{Unblind}(C''_{t_{eph_j}}),\ M = \mathsf{Decrypt}_0(C_m, K).$$

With respect to the efficiency, this protocol may be quite inefficient in practice. The main reason is that the Ephemerizer potentially needs to publish and certify all the ephemeral public keys before data can be encrypted by the user. Considering the fact that the data may have a wide range of expiration time, the Ephemerizer may need to publish a large volume of key pairs.

3.1.2. *Security Analysis of the Protocol*

In [3, 4], the following assumptions are made on the validation of public keys.

1. The user should validate that the ephemeral public keys $PK_{t_{eph_j}}$ is certified by a long-term private key of the Ephemerizer, where the corresponding long-term public key is certified by a Trusted Third Party (TTP).
2. There is no need for the Ephemerizer and the user to authenticate each other. There is no need to encrypt or protect the integrity of the ephemeral key sent to the user, i.e. there is no need for the user to check the validity of $PK_{t_{eph_j}}$ in any received message $(KeyID_{t_{eph_j}}, C, PK_{t_{eph_j}})$.

We show below that lacking of validation of $PK_{t_{eph_j}}$ by the user may lead to a potential security vulnerability if the Ephemerizer is curious. As one of the options suggested in [3, 4], we suppose that the public key encryption scheme $\mathcal{E}_1$ is RSA [12]. Then, the above Ephemerizer protocol will be instantiated to be the following.

- $\mathsf{Setup}_E(\ell)$: The Ephemerizer generates $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ in the form $((e_j, N_j), d_j)$ where $e_j d_j \equiv 1 \pmod{\varphi(N_j)}$.
- $\mathsf{Setup}_U(\ell)$: The algorithm is the same as in the above.
- $\mathsf{Encrypt}(M, PK_U, PK_{t_{eph_j}})$: The ciphertext is $(KeyID_{t_{eph_j}}, C, (e_j, N_j))$, where $C = (C_m, C_{t_{eph_j}})$,

$$C_m = \mathsf{Encrypt}_0(M, K),$$

$$C_k = K^{e_j} \mod N_j,\ C_{t_{eph_j}} = \mathsf{Encrypt}_2(C_k, PK_U).$$

- $\mathsf{Decrypt}(C, SK_U; SK_{t_{eph_j}})$:
  1. The user generates $R \in_R \mathbb{Z}^*_{N_j}$
  2. The user decrypts $C_{t_{eph_j}}$ to obtain $C_k = K^{e_j} \mod N_j$, and then computes and sends $(KeyID_{t_{eph_j}}, C'_{t_{eph_j}})$ to the Ephemerizer, where
  $$R \in_R \mathbb{Z}^*_N,\ C'_{t_{eph_j}} = K^{e_j} R^{e_j} \mod N_j. \qquad (1)$$
  3. If the ephemeral key $SK_{t_{eph_j}}$ associated with $KeyID_{t_{eph_j}}$ has not expired, the Ephemerizer decrypts $C'_{t_{eph_j}}$ and sends $C''_{t_{eph_j}}$ to the user, where
  $$\begin{aligned} C''_{t_{eph_j}} &= (C'_{t_{eph_j}})^{d_j} \mod N_j \\ &= (K^{e_j} R^{e_j})^{d_j} \mod N_j \\ &= KR \mod N_j. \end{aligned}$$
  4. The user obtain $K = C''_{t_{eph_j}} R^{-1} \mod N_j$, and then decrypts C_m to obtain $M = \mathsf{Decrypt}_0(C_m, K)$.

Suppose the Ephemerizer has obtained $(KeyID_{t_{eph_j}}, C, (e_j, N_j))$ by eavesdropping on the user's communications. In order to recover the key K, the Ephemerizer can send $(KeyID_{t_{eph_j}}, C, (\varphi(N_j), N_j))$ to the user. Note that the Ephemerizer knows $\varphi(N_j)$. According to the $\mathsf{Decrypt}$ algorithm, the user will send $(KeyID_{t_{eph_j}}, C'_{t_{eph_j}})$, where

$$\begin{aligned} C'_{t_{eph_j}} &= K^{e_j} R^{\varphi(N_j)} \mod N_j \\ &= K^{e_j} \mod N_j, \end{aligned}$$

to the Ephemerizer for blind decryption. Clearly, the Ephemerizer can obtain $K = (C'_{t_{eph_j}})^{d_j} \mod N_j$.

Due to the fact that the ephemeral public keys are only required to be certified by the private key of the Ephemerizer, the presented security vulnerability will still remain even if the user validates the public keys in the ciphertext. One possible countermeasure is to let a TTP to certify that the ephemeral public keys of Ephemerizer are generated properly.

3.2. The Hybrid PKI-IBC Ephemerizer Protocol

3.2.1. *Description of the Protocol*

The Ephemerizer protocol [5] consists of algorithms $(\mathsf{Setup}_E, \mathsf{Setup}_U, \mathsf{Encrypt}, \mathsf{Decrypt})$, which are defined as follows.

- $\mathsf{Setup}_E(\ell)$: The Ephemerizer first generates a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$, a generator $g \in_R \mathbb{G}$, where $\mathbb{G}$ and $\mathbb{G}_1$ are multiplicative groups of prime order p. The Ephemerizer then generates a long-term private key $SK_E \in_R \mathbb{Z}_p$ and the public key $PK_E = g^{SK_E}$, two hash functions
$$\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G},\ \mathsf{H}_2 : \mathbb{G}_1 \rightarrow \{0,1\}^n,$$
and a set of ephemeral tuples $(KeyID_{t_{eph_j}}, PK_{t_{eph_j}}, SK_{t_{eph_j}}, t_{exp_j})$ where $1 \le j \le \mathcal{N}$, $KeyID_{t_{eph_j}}$ is the identifier of this tuple, and $PK_{t_{eph_j}} = g^{SK_{t_{eph_j}}}$. Suppose also that the Ephemerizer possesses the identity ID_E.
- $\mathsf{Setup}_U(\ell)$: The user generates a key pair (PK_U, SK_U) for a public key encryption scheme $\mathcal{E}_2$ with algorithms $(\mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$. The user also selects a symmetric key encryption scheme
$$\mathcal{E}_1 = (\mathsf{Encrypt}_1, \mathsf{Decrypt}_1),$$
which has the key space $\mathcal{K}$ and will be used to encrypt data in the system.
- $\mathsf{Encrypt}(M, PK_U, PK_{t_{eph_j}})$: The ciphertext is $(KeyID_{t_{eph_j}}, C)$, where $r \in_R \mathbb{Z}_p$, $K \in_R \mathcal{K}$,
$$C_m = \mathsf{Encrypt}_1(M, K),\ C_k = \mathsf{Encrypt}_2(K, PK_U), \quad (2)$$
$$ID_{t_{eph_j}} = ID_E \| Expiry : t'_{exp_j},$$
$$Q_{t_{eph_j}} = \hat{e}(\mathsf{H}_1(ID_{t_{eph_j}}), PK_{t_{eph_j}}),$$
$$C_{t_{eph_j}} = (g^r, C_k \oplus \mathsf{H}_2((Q_{t_{eph_j}})^r)), \quad (3)$$
$$C^{\dagger}_{t_{eph_j}} = \mathsf{Encrypt}_2(ID_{t_{eph_j}} \| C_{t_{eph_j}}, PK_U), \quad (4)$$
$$C = (C_m, C^{\dagger}_{t_{eph_j}})$$
It is required t'_{exp_j} should be smaller than t_{exp_j} which is the expiration time of $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$.
- $\mathsf{Decrypt}(C, SK_U; SK_{t_{eph_j}}, SK_E)$:
  1. The user 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
  $$ID'_{t_{eph_j}} \in_R \{0,1\}^*,\ Q'_{t_{eph_j}} = \hat{e}(\mathsf{H}_1(ID'_{t_{eph_j}}), PK_E),$$
  $$r' \in_R \mathbb{Z}_p,\ K' \in_R \mathcal{K},$$
  $$C'_{t_{eph_j}} = (g^{r'}, (ID_{t_{eph_j}} \| K') \oplus \mathsf{H}_2((Q'_{t_{eph_j}})^{r'})). \quad (5)$$
  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
  $$ID_{t_{eph_j}} \| K' = (ID_{t_{eph_j}} \| K') \oplus \mathsf{H}_2((Q'_{t_{eph_j}})^{r'}) \oplus \mathsf{H}_2(\hat{e}(\mathsf{H}_1(ID'_{t_{eph_j}}), g^{r'})^{SK_E}). \quad (6)$$
  It then computes and sends $C''_{t_{eph_j}}$ to the user, where
  $$C''_{t_{eph_j}} = \mathsf{Encrypt}_1(\mathsf{H}_1(ID_{t_{eph_j}})^{SK_{t_{eph_j}}}, K'). \quad (7)$$
  3. The user decrypts $C''_{t_{eph_j}}$ to obtain $\mathsf{H}_1(ID_{t_{eph_j}})^{SK_{t_{eph_j}}}$, and then decrypts $C_{t_{eph_j}}$ to obtain C_k as follows.
  $$C_k = C_k \oplus \mathsf{H}_2((Q_{t_{eph_j}})^{r}) \oplus \mathsf{H}_2(\hat{e}(g^{r}, \mathsf{H}_1(ID_{t_{eph_j}})^{SK_{t_{eph_j}}})). \quad (8)$$
  The user then sequentially decrypts C_k and C_m to obtain M as follows:
  $$K = \mathsf{Decrypt}_2(C_k, SK_U),\ M = \mathsf{Decrypt}_1(C_m, K). \quad (9)$$

3.2.2. *Security Analysis of the Protocol*

On the exact expiration time. In the above protocol, when the entity, who runs $\mathsf{Encrypt}$, constructs $ID_{t_{eph_j}} = ID_E \| Expiry : t'_{exp_j}$, it 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 user, then it 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}.

On recovering expired data. In [5], no rigorous analysis has been done for this protocol. Next, we show that expired data can still be recovered by an adversary. Suppose that, through eavesdropping, the adversary has obtained $(C, C'_{t_{eph_j}}, C''_{t_{eph_j}})$, where

$$C = (C_m, C^{\dagger}_{t_{eph_j}}),\ C_m = \mathsf{Encrypt}_1(M, K),$$

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

$$C'_{t_{eph_j}} = (g^{r'}, (ID_{t_{eph_j}} \| K') \oplus \mathsf{H}_2((Q'_{t_{eph_j}})^{r'})),$$

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

Suppose that, at the time $t_{eph_{j+1}}$, where $t_{eph_{j+1}} > t_{eph_j}$, the adversary compromises the Ephemerizer and the user, and obtains SK_E and SK_U. Note that, at the time $t_{eph_{j+1}}$, $SK_{t_{eph_j}}$ has been securely deleted and any ciphertext encrypted with $PK_{t_{eph_j}}$ should be unrecoverable.

1. Based on the equation (5), using SK_E, the adversary can decrypt $C'_{t_{eph_j}}$ and obtain $ID_{t_{eph_j}} \| K'$.
2. Based on the equation (7), using K', the adversary can recover $\mathsf{H}_1(ID_{t_{eph_j}})^{SK_{t_{eph_j}}}$ by decrypting $C''_{t_{eph_j}}$.
3. Based on the equation (4), using SK_U, the adversary can recover $C_{t_{eph_j}}$ by decrypting C.
4. Based on the equation (3), using $\mathsf{H}_1(ID_{t_{eph_j}})^{SK_{t_{eph_j}}}$, the adversary can recover C_k by decrypting $C_{t_{eph_j}}$.
5. Based on the equations (2), using SK_U, the adversary can recover K by decrypting C_k, and then recover M by decrypting C_m using K.

4. THE CONCEPT OF TIMED-EPHEMERIZER

A Timed-Ephemerizer protocol guarantees that encrypted data will only be recoverable 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 [5, 3, 4], a Timed-Ephemerizer protocol explicitly provides the guarantee that data can only be available after the pre-defined initial disclosure time.

Generally, a Timed-Ephemerizer protocol involves the following types of entities: a time server, users, and an Ephemerizer. 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 guarantee 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 one of them is compromised.

The idea behind of this new primitive is similar to that of Ephemerizer. The Ephemerizer plays the same role as in the case of Ephemerizer and generates a set of ephemeral key pairs $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ $(1 \leq j \leq \mathcal{N})$. The time server defines a time set $\mathcal{T}$ and sequentially publishes a timestamp TS_t at the time $t \in \mathcal{T}$. Data is encrypted using a symmetric key, which will be encrypted using one ephemeral public key of the Ephemerizer and the public key of the user while taking into account an initial disclosure time $t_{int} \in \mathcal{T}$. The decryption is an interactive algorithm between the user and the Emphemerizer. The security is mainly based on two facts.

1. Only at time $t \in \mathcal{T}$, the time server publishes a timestamp TS_t, so that any ciphertext generated with time t will only become recoverable after t.
2. At the time t_{eph_j}, the Ephemerizer will delete $SK_{t_{eph_j}}$, so that any ciphertext generated under the ephemeral public key $PK_{t_{eph_j}}$ will become unrecoverable afterwards.

In the rest of this section, we define the algorithms of Timed-Ephemerizer and formalize its security properties.

4.1. The Algorithm Definitions

A Timed-Ephemerizer protocol consists of the following algorithms. Note that, compared with an Ephemerizer protocol, there are two additional algorithms, namely Setup_T and $\mathsf{TimeExt}$.

- $\mathsf{Setup}_T(\ell)$: Run by the time server, this algorithm generates a public/private key pair (PK_T, SK_T). In addition, the time server also publishes a time set $\mathcal{T}$.
- $\mathsf{TimeExt}(t, SK_T)$: Run by the time server, this algorithm generates a timestamp TS_t for $t \in \mathcal{T}$. It is assumed that the time server publishes TS_t at the point t.
- $\mathsf{Setup}_E(\ell)$: Run by the Ephemerizer, this algorithm generates a set of tuples $(PK_{t_{eph_j}}, SK_{t_{eph_j}}, t_{eph_j})$, where $1 \leq j \leq \mathcal{N}$, $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ is an ephemeral public/private key pair, and t_{eph_j} is the expiration time. 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}.
- $\mathsf{Setup}_U(\ell)$: Run by a user, this algorithm generates a public/private key pair (PK_U, SK_U).
- $\mathsf{Encrypt}(M, t_{int}, PK_U, PK_{t_{eph_j}}, PK_T)$: This algorithm outputs a ciphertext C. For the message M, $t_{int} \in \mathcal{T}$ 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 user.
- $\mathsf{Decrypt}(C, TS_{t_{int}}, SK_U; SK_{t_{eph_j}})$: Run between the user and the Ephemerizer, this algorithm outputs a plaintext M or an error symbol for the user.

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. In the algorithm designs, we let the time server and the Ephemerizer choose $\mathcal{T}$ and t_{eph_j} $(1 \leq j \leq \mathcal{N})$ respectively. How to choose these timestamps is a practical issue, and may be subject to the application environment where a Timed-Ephemerizer protocol will be used. However, no matter how this will be done, it will not affect our security analysis.

Note that a Timed-Ephemerizer protocol can be employed by an entity, say Alice, to protect her own data in persistent storage devices or protect her data that she wants to share with another entity, say Bob. In the first situation, Alice encrypts her data $\mathsf{Encrypt}(M, t_{int}, PK_A, PK_{t_{eph_j}}, PK_T)$, where PK_A is Alice's public key. In the second situation, Alice encrypts her data $\mathsf{Encrypt}(M, t_{int}, PK_B, PK_{t_{eph_j}}, PK_T)$, where PK_B is Bob's public key. The example in Section 6.2 is in the second situation.

Similar to other cryptographic primitives, the basic requirement to Timed-Ephemerizer is soundness. Informally, this property means that the algorithms Encrypt and Decrypt work properly with valid inputs. Formally, it is defined as follows.

Definition 4.1. *A Timed-Ephemerizer protocol achieves (unconditional) soundness if the following equality always holds for any M, t_{int}, j.*

$$\mathsf{Decrypt}(\mathsf{Encrypt}(M, t_{int}, PK_U, PK_{t_{eph_j}}, PK_T), TS_{t_{int}}, SK_U; SK_{t_{eph_j}}) = M.$$

4.2. The Security Definitions

With respect to the entities involved in a Timed-Ephemerizer protocol, we make the following trust assumptions.

- The time server publishes TS_t at the time t, where $t \in \mathcal{T}$. Except for this, the time server may try to obtain some information about users' plaintext data.
- A user is supposed to access the data during its lifecycle, but not beyond it. Furthermore, the user will not store plaintext data on persistent storage devices.
- The Ephemerizer publishes ephemeral public/private key pairs and revoke them periodically. Except for this, the time server may try to obtain some information about users' plaintext data.

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. For example, the validation can be done with the help of a TTP. Note that the validation of keys are crucial to achieve the security properties against both Type-I and Type-II adversaries. 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 of data, 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 user and also a malicious outside entity which has compromised the Ephemerizer and the user 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 user after the expiration time of the data.
- Type-III adversary: This type of adversary represents a malicious time server and a malicious Ephemerizer, and also a malicious outside entity which has compromised the time server and the Ephemerizer.

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 user's private key.

It is worth stressing that when the adversary compromises an entity (the time server, the Ephemerizer, or the user) 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 4.2. *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 Fig. 2), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

In the attack game, PK_* means all available public keys. 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. $(PK_T, SK_T) \xleftarrow{\$} \mathsf{Setup}_T(\ell)$;
   $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $1 \le j \le \mathcal{N} \xleftarrow{\$} \mathsf{Setup}_E(\ell)$;
   $(PK_U, SK_U) \xleftarrow{\$} \mathsf{Setup}_U(\ell)$
2. $(M_0, M_1, t^*_{int}, PK_{t_{eph_i}}) \xleftarrow{\$} \mathcal{A}^{(\mathsf{TimeExt})}(SK_{t_{eph_j}} \ (1 \le j \le \mathcal{N}), SK_U, PK_*)$
3. $b \xleftarrow{\$} \{0,1\}; C_b \xleftarrow{\$} \mathsf{Encrypt}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T)$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{TimeExt})}(C_b, SK_{t_{eph_j}}$ for $1 \le j \le \mathcal{N}, SK_U, PK_*)$
```

FIGURE 2. Semantic Security against Type-I Adversary

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

FIGURE 3. Semantic Security against Type-II Adversary

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 $1 \le j \le \mathcal{N}$, 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.
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}) where $t^*_{int} \in \mathcal{T}$ and $1 \le i \le \mathcal{N}$. The only restriction is that the TimeExt oracle should not have been queried with $t \ge t^*_{int}$.
3. The challenger picks a random bit $b \in \{0,1\}$ and gives the adversary C_b as the challenge, where
$$C_b = \mathsf{Encrypt}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T).$$
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 $1 \le j \le \mathcal{N}$. The restriction in steps 2 and 4 of the above game, namely "the TimeExt oracle should not have been queried with $t \ge t^*_{int}$.", implies that the adversary tries to recover a message before the initial disclosure time. This coincides with the definition of Type-I adversary.

Definition 4.3. *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 Fig. 3), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

In the attack game, PK_* means all available public keys. 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 user.

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 $1 \le j \le \mathcal{N}$, 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 Decrypt oracle queries.
   (a) D-type Decrypt 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 Decrypt 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 Decrypt query: In each oracle query, the adversary impersonates a user to the Ephemerizer and sends t_{eph_j} to the challenger, which uses $SK_{t_{eph_j}}$ as the input and runs the Decrypt 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}) where $t^*_{int} \in \mathcal{T}$ and $1 \le i \le \mathcal{N}$. In this phase, the adversary can query for SK_U and $SK_{t_{eph_j}}$ for any $i < j \le \mathcal{N}$ with the following restriction: if SK_U has been queried, then any E-type Decrypt oracle query with the input t_{eph_j} for any $1 \le j \le i$ is forbidden.
3. The challenger picks a random bit $b \in \{0,1\}$ and gives the adversary C_b as the challenge, where
$$C_b = \mathsf{Encrypt}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T).$$
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 $i < j \le \mathcal{N}$. In the above game, the privilege, that the adversary can issue the two types of Decrypt oracle queries, reflects the fact that the adversary has complete control over the communication link between the user and the Ephemerizer. In practice, such an adversary can initiate the Decrypt algorithm with both the Ephemerizer and the user. The first case is modeled by the E-type Decrypt query, while the second case is modeled by the D-type Decrypt query. The restriction in the above game, namely "if SK_U has been queried, then E-type Decrypt oracle query with the input t_{eph_j} for any $1 \le j \le i$ 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 $1 \le j \le i$ should have been securely deleted by the Ephemerizer). This coincides with the definition of Type-II adversary.

Definition 4.4. *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 Fig. 4), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

1. $(PK_T, SK_T) \overset{\$}{\leftarrow} \mathcal{A}(\ell); (PK_{t_{eph_j}}, SK_{t_{eph_j}})for1 \le j \le \mathcal{N} \overset{\$}{\leftarrow} \mathcal{A}(\ell); (PK_U, SK_U) \overset{\$}{\leftarrow} \mathsf{Setup}_U(\ell)$
2. $(M_0, M_1, t^*_{int}, PK_{t_{eph_i}}) \overset{\$}{\leftarrow} \mathcal{A}^{(\mathsf{Decrypt})}(SK_T, SK_{t_{eph_j}}$ for $1 \le j \le \mathcal{N}, PK_*)$
3. $b \overset{\$}{\leftarrow} \{0,1\}; C_b \overset{\$}{\leftarrow} \mathsf{Encrypt}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T)$
4. $b' \overset{\$}{\leftarrow} \mathcal{A}^{(\mathsf{Decrypt})}(C_b, SK_T, SK_{t_{eph_j}}$ for $1 \le j \le \mathcal{N}, PK_*)$

FIGURE 4. Semantic Security against Type-III Adversary

In the attack game, PK_* means all available public keys. 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 user.

1. The adversary $\mathcal{A}$ generates (PK_T, SK_T) and $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $1 \le j \le \mathcal{N}$. Note that the adversary may not follow the protocol specification to generate these parameters. The challenger runs Setup_U to generate (PK_U, SK_U). The public key PK_U is given to the adversary.
2. The adversary can adaptively issue the D-type Decrypt 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}) where $t^*_{int} \in \mathcal{T}$ and $1 \le i \le \mathcal{N}$.
3. The challenger picks a random bit $b \in \{0,1\}$ and gives the adversary C_b as the challenge, where
$$C_b = \mathsf{Encrypt}(M_b, t^*_{int}, PK_U, PK_{t_{eph_i}}, PK_T).$$
4. The adversary can continue to query the Decrypt 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 $1 \le j \le \mathcal{N}$. In the above game, expect for the user's private key, the adversary is allowed to access all other secrets. In particular, this means that an outside adversary can compromise both the time server and the Ephemerizer at any time. This coincides with the definition of Type-III adversary.

5. A NEW TIMED-EPHEMERIZER PROTOCOL

In this section, we propose revise the Timed-Ephemerizer protocol from [7] and prove its security. The philosophy behind the proposed protocol is similar to the blind decryption technique [3, 4]. As in the case of the hybrid PKI-IBC protocol [5], the proposed protocol also adopts the concept of identity-based encryption [9, 13] to avoid publishing a large volume of ephemeral public keys. Here are some intuitions about the construction.

1. *Encryption.* Data is first encrypted jointly using the ephemeral public key of the Ephemerizer and the public key of the time server. This is illustrated in the generation of the element C_4 in the ciphertext, referring to the equation (10). The ciphertext is then re-encrypted using the public key of the user. The $\oplus$ operation in generating C_4 allows the re-randomization in the decryption, while the checksum C_7 is important to achieve the security properties against Type-I and Type-II adversaries (technically, allowing us to answer the adversaries' oracle queries in the proofs).
2. *Decryption.* In order to achieve the security property against Type-III adversary, the ciphertext need to be re-randomized in the decryption algorithm. As a result, in decryption, the ciphertext is firstly decrypted using the user's private key, and then re-randomized using a random number. This is illustrated in the generation of the element C'_4 in the ciphertext, referring to the equation (12). The re-randomized data is then encrypted using an ephemeral public key of the Ephemerizer and sent to the the Ephemerizer. This additional encryption process is used to prevent replay attacks from Type-I and Type-II adversaries, due to the fact that the re-randomization is simply an XOR operation.

5.1. The Proposed Construction

Let $\{0,1\}^n$ be the message space of user, 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 $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$, a cryptographic hash function H_1, and a long-term public/private key pair (PK_T, SK_T) where

$$\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G},\ SK_T \in_R \mathbb{Z}_p,\ PK_T = g^{SK_T}.$$

The time server also publishes $(\mathbb{G}, \mathbb{G}_1, p, g, \hat{e}, \mathsf{H}_1)$. Suppose that the time server possesses the identity ID_T. In addition, the time server also publishes a time set $\mathcal{T}$. Since the details how to define $\mathcal{T}$ will not affect out analysis, we keep it at an abstract level.

- $\mathsf{TimeExt}(t, SK_T)$: 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} $(1 \le j \le \mathcal{N})$ where $\mathcal{N}$ is an integer. The Ephemerizer generates a master key pair $(PK_E^{(0)}, SK_E^{(0)})$ and a hash function H_2, where

$$SK_E^{(0)} \in_R \mathbb{Z}_p,\ PK_E^{(0)} = g^{SK_E^{(0)}},\ \mathsf{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^n,$$

and sets, for $1 \le j \le \mathcal{N}$,

$$PK^{(0)}_{t_{eph_j}} = ID_E \| t_{eph_j},\ SK^{(0)}_{t_{eph_j}} = \mathsf{H}_1(ID_E \| t_{eph_j})^{SK_E^{(0)}}.$$

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, for $1 \le j \le \mathcal{N}$, generates the ephemeral key pairs

$$(PK^{(1)}_{t_{eph_j}},\ SK^{(1)}_{t_{eph_j}}),\ \text{where}\ PK^{(1)}_{t_{eph_j}} = ID_E \| t_{eph_j}.$$

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 $1 \le j \le \mathcal{N}$, where

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

In addition, the Ephemerizer deletes $SK_E^{(0)}, SK_E^{(1)}$ and 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 user publishes the following hash functions.

$$\mathsf{H}_3 : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G},\ \mathsf{H}_4 : \mathcal{Y} \rightarrow \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^n,$$

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

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

$$r_1, r_2 \in_R \mathbb{Z}_p, X \in_R \mathcal{X}, C_1 = g^{r_1}, C_2 = g^{r_2}, C_3 = \mathsf{H}_3(C_1 \| C_2)^{r_1},$$

$$\begin{aligned} C_4 &= M \oplus \mathsf{H}_2(\hat{e}(\mathsf{H}_1(PK^{(0)}_{t_{eph_j}}), PK_E^{(0)})^{r_1} \cdot \hat{e}(\mathsf{H}_1(ID_T \| t_{int}), PK_T)^{r_2}) \quad (10) \\ &= M \oplus \mathsf{H}_2(\hat{e}(\mathsf{H}_1(ID_E \| t_{eph_j}), C_1)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T \| t_{int}), C_2)^{SK_T}) \quad (11) \end{aligned}$$

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

$$C_7 = \mathsf{H}_2(X \| C_1 \| C_2 \| C_3 \| C_4 \| C_5 \| C_6),\ C = (C_5, C_6, C_7).$$

- $\mathsf{Decrypt}(C, TS_{t_{int}}, SK_U; SK_{t_{eph_j}})$:
  1. The user decrypts C_5 to obtain X, and aborts if the following inequality is true.

     $$C_7 \neq \mathsf{H}_2(X \| (C_6 \oplus \mathsf{H}_5(X)) \| C_5 \| C_6)$$

     Otherwise it computes $C_1 \| C_2 \| C_3 \| C_4 = \mathsf{H}_5(X) \oplus C_6$. The user then retrieves $TS_{t_{int}}$ from the time server and checks that $\hat{e}(TS_{t_{int}}, g) = \hat{e}(\mathsf{H}_1(ID_T \| t_{int}), PK_T)$. If so, it computes and sends $(C', TS_{t_{int}})$ to the Ephemerizer, where $M' \in_R \{0,1\}^n$,

     $$C'_1 = C_1, C'_2 = C_2, C'_3 = C_3, C'_4 = M' \oplus C_4, \quad (12)$$

     $$Y \in_R \mathcal{Y},\ C'_5 = \mathsf{Encrypt}_1(Y, PK^{(1)}_{t_{eph_j}}),$$

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

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

     $$C' = (C'_5, C'_6, C'_7).$$

  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

     $$C'_7 \neq \mathsf{H}_2(Y \| (C'_6 \oplus \mathsf{H}_4(Y)) \| C'_5 \| C'_6).$$

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

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

     Finally, it sends C'' to the user, where

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

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

Since identity-based encryption is used, only the long-term public key $PK_E = (PK_E^{(0)}, PK_E^{(1)})$ needs to be certified by a TTP. Instead of publishing and certifying all the ephemeral public keys, as in the case of [3, 4], the Ephemerizer only needs to publish t_{eph_j}

$(1 \leq j \leq \mathcal{N})$. Compared with the protocol in [5], the concrete difference is that the master private key $SK_E = (SK_E^{(0)}, SK_E^{(1)})$ is only required to be ephemeral, i.e. after generating the ephemeral private keys, the Ephemerizer can delete SK_E.

It is straightforward to verify that the soundness property is achieved, namely the Encrypt and Decrypt work properly. We skip the details here.

5.2. The Security Analysis

In the execution of Decrypt, the timestamp $TS_{t_{int}}$ is a required input. Intuitively, before the time server publishes the timestamp, it is infeasible for the user and the Ephemerizer to run Decrypt to recover the message. The following lemma formalizes this intuition.

Lemma 5.1. *The proposed scheme achieves semantic security against Type-I adversary based on the BDH assumption in the random oracle model.*

Proof sketch. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the attack game depicted in Figure 2.

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 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.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^* \in_R \mathcal{X},\ R \in \mathbb{G}_1,\ C_1^* = g^{r_1^*},\ C_2^* = g^{r_2^*},$$

$$C_3^* = \mathsf{H}_3(C_1^*||C_2^*)^{r_1^*}, C_4^* = M_b \oplus \mathsf{H}_2(R),$$

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

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

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_2 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 $\hat{e}(\mathsf{H}_1(ID_E||t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T||t_{int}^*), C_2^*)^{SK_T}$ has been replaced with R, where $R \in_R \mathbb{G}_1$. As a result, Game_1 is identical to Game_0 unless $\hat{e}(\mathsf{H}_1(ID_E||t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T||t_{int}^*), C_2^*)^{SK_T}$ has been queried to H_2. Note that SK_T is not required in answering the TimeExt oracle queries. We immediately obtain $|\delta_1 - \delta_0| = \epsilon'$ where ϵ' is negligible based on the BDH assumption. The lemma now follows. □

Lemma 5.2. *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.*

Proof sketch. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the attack game depicted in Figure 3. The security proof is done through a sequence of games [14].

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.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^* \in_R \mathcal{X},$$

$$C_1^* = g^{r_1^*},\ C_2^* = g^{r_2^*},\ C_3^* = \mathsf{H}_3(C_1^*||C_2^*)^{r_1^*},$$

$$C_4^* = M_b \oplus \mathsf{H}_2(\hat{e}(\mathsf{H}_1(ID_E||t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T||t_{int}^*), C_2^*)^{SK_T}),$$

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

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

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. Before the adversary queries SK_U, given a D-type Decrypt query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$, the challenger answers as the following.

1. In step 4 of the game, if $C = C_b$, the challenger returns C', where

$$M' \in_R \{0,1\}^n,\ C_1' = C_1^*,\ C_2' = C_2^*,\ C_3' = C_3^*,$$

$$C_4' = M' \oplus C_4^*,\ Y \in_R \mathcal{Y},\ C_5' = \mathsf{Encrypt}_1(Y, PK_{t_{eph_j}}^{(1)}),$$

$$C_6' = \mathsf{H}_4(Y) \oplus (C_1'||C_2'||C_3'||C_4'),$$

$$C_7' = \mathsf{H}_2(Y||C_1'||C_2'||C_3'||C_4'||C_5'||C_6'),\ C' = (C_5', C_6', C_7').$$

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_2 such that

$$C_5 = \mathsf{Encrypt}_2(\tilde{X}, PK_U),\ C_6 = \tilde{C}_6, \quad (13)$$

$$\mathsf{H}_5(\tilde{X}) \oplus C_6 = \tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4,\ C_7 = \mathsf{H}_2(\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6). \quad (14)$$

If the input exists, the challenger returns C', where

$$M' \in_R \{0,1\}^n,\ C'_1 = \tilde{C}_1,\ C'_2 = \tilde{C}_2,\ C'_3 = \tilde{C}_3,$$

$$C'_4 = M' \oplus \tilde{C}_4, C'_5 = \mathsf{Encrypt}_1(Y, PK^{(1)}_{t_{eph_j}}),$$

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

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

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 $\mathsf{Decrypt}$ oracle queries.

- In the first case, there is a query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$ such that an oracle query to H_2 with the input $\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6$ (these values are determined by the equalities (13) and (14)) returns C_7, while the C_7 is chosen before the oracle query is made. Or,
- In the second case, there is a query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$ such that oracle queries to H_2 with different inputs $\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6$ return C_7.

As H_2 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 $|\delta_1 - \delta_0| \le \epsilon_1 = \Pr[Evn]$ is negligible.

Before moving forward, we first describe the following claim. The verification of this claim can be done straightforwardly in the random oracle model given the encryption schemes $\mathcal{E}_1$ and $\mathcal{E}_2$ are one-way permutations.

Claim 1. Before the adversary queries SK_U, given an E-type $\mathsf{Decrypt}$ 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 $\mathsf{Decrypt}$ query, then the probability $C'_1 = C^*_1$ is negligible, where $Y = \mathsf{Decrypt}_1(C'_5, SK^{(1)}_{t_{eph_i}})$ and $C'_1||C'_2||C'_3||C'_4 = \mathsf{H}_4(Y) \oplus C'_6$.

Game_2: In this game, the challenger performs in the same way as in Game_1 except for the following. Before the adversary queries SK_U, for any E-type $\mathsf{Decrypt}$ query with the input $(C' = (C'_5, C'_6, C'_7), TS_{t_{int}}, t_{eph_i})$, the challenger rejects the request if $C'_1 = C^*_1$, where $Y = \mathsf{Decrypt}_1(C'_5, SK^{(1)}_{t_{eph_i}})$ and $C'_1||C'_2||C'_3||C'_4 = \mathsf{H}_4(Y) \oplus C'_6$, and C' is not one of the output of D-type $\mathsf{Decrypt}$ queries.

Let δ_2 be the probability that the challenger successfully ends and $b' = b$ in Game_2. From the above claim, we have $|\delta_2 - \delta_1| = \epsilon_2$ is negligible.

Game_3: In this game, the challenger performs in the same way as in Game_2 except for the following. Before the adversary queries SK_U, for any E-type $\mathsf{Decrypt}$ query with the input $(C' = (C'_5, C'_6, C'_7), TS_{t_{int}}, t_{eph_i})$, the challenger returns $T \in_R \{0,1\}^n$ if $C'_1 = C^*_1$ where $Y = \mathsf{Decrypt}_1(C'_5, SK^{(1)}_{t_{eph_i}})$ and $C'_1||C'_2||C'_3||C'_4 = \mathsf{H}_4(Y) \oplus C'_6$, and C' is one of the output of D-type $\mathsf{Decrypt}$ queries.

The game Game_3 is identical to Game_2 unless the following event Evn occurs: For some aforementioned E-type $\mathsf{Decrypt}$ oracle query with the input $(C' = (C'_5, C'_6, C'_7), TS_{t_{int}}, t_{eph_i})$, the adversary has queried H_2 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_3 be the probability that the challenger successfully ends and $b' = b$ in Game_3. Therefore, we have $|\delta_3 - \delta_2| \le \epsilon_3 = \Pr[Evn]$ is negligible.

Game_4: In this game, the challenger performs in the same way as in Game_3 except for answering the $\mathsf{Decrypt}$ oracle queries.

- Before the adversary queries SK_U, given an E-type $\mathsf{Decrypt}$ query with the input $(C' = (C'_5, C'_6, C'_7), TS_{t_{int}}, t_{eph_i})$, if $C'_1 \neq C^*_1$ where $Y = \mathsf{Decrypt}_1(C'_5, SK^{(1)}_{t_{eph_i}})$ and $C'_1||C'_2||C'_3||C'_4 = \mathsf{H}_4(Y) \oplus C'_6$, the challenger first 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_2 such that
$$C'_5 = \mathsf{Encrypt}_1(\tilde{Y}, PK^{(1)}_{t_{eph_i}}),\ C'_6 = \tilde{C}'_6, \tag{15}$$
$$\mathsf{H}_4(\tilde{Y}) \oplus C'_6 = \tilde{C}'_1||\tilde{C}'_2||\tilde{C}'_3||\tilde{C}'_4, C'_7 = \mathsf{H}_2(\tilde{Y}||\tilde{C}'_1||\tilde{C}'_2||\tilde{C}'_3||\tilde{C}'_4||\tilde{C}'_5||\tilde{C}'_6). \tag{16}$$
If the input exists, the challenger proceeds. If $\hat{e}(\tilde{C}'_3, g) \neq \hat{e}(\tilde{C}'_1, \mathsf{H}_3(\tilde{C}'_1||\tilde{C}'_2))$, it aborts; otherwise it returns C'', where
$$C'' = \mathsf{H}_2(\tilde{Y}||\tilde{C}'_1||\tilde{C}'_2||\tilde{C}'_3||\tilde{C}'_4) \oplus \tilde{C}'_4 \oplus \mathsf{H}_2(\hat{e}(PK^{(0)}_E, \mathsf{H}_1(PK^{(0)}_{t_{eph_i}}))^{\tilde{r}'_1} \cdot \hat{e}(TS_{t_{int}}, \tilde{C}'_2)).$$
Note that the challenger retrieves $\tilde{r}'_1$ such that $g^{\tilde{r}'_1} = \tilde{C}'_1$.

Let δ_4 be the probability that the challenger successfully ends and $b' = b$ in Game_4. The game Game_4 is identical to Game_3 unless the following event Evn occurs in answering the E-type $\mathsf{Decrypt}$ oracle queries.

- In the second case, there is a query with the input $(C' = (C'_5, C'_6, C'_7), TS_{t_{int}}, t_{eph_i})$ such that an oracle query to H_2 with the input $\tilde{Y}||\tilde{C}'_1||\tilde{C}'_2||\tilde{C}'_3||\tilde{C}'_4||\tilde{C}'_5||\tilde{C}'_6$ (these values are determined by the equalities (15) and (16)) returns C'_7, while the C'_7 is chosen before the oracle query is made. Or,
- In the second case, there is a query with the input $(C' = (C'_5, C'_6, C'_7), TS_{t_{int}}, t_{eph_i})$ such that oracle queries to H_2 with different inputs $\tilde{Y}||\tilde{C}'_1||\tilde{C}'_2||\tilde{C}'_3||\tilde{C}'_4||\tilde{C}'_5||\tilde{C}'_6$ return C'_7.

As H_2 is modeled as a random oracle, the probability $\Pr[Evn]$ is negligible. Let δ_4 be the probability that the challenger successfully ends and $b' = b$ in Game_4. Therefore, we have $|\delta_4 - \delta_3| \le \epsilon_4 = \Pr[Evn]$ is negligible.

Game_5: In this game, the challenger performs in the same way as in Game_4 except that the challenge C_b is computed as follows.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^* \in_R \mathcal{X},\ R \in \mathbb{G}_1,\ C_1^* = g^{r_1^*},$$

$$C_2^* = g^{r_2^*},\ C_3^* = \mathsf{H}_3(C_1^*||C_2^*)^{r_1^*},\ C_4^* = M_b \oplus \mathsf{H}_2(R),$$

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

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

Let δ_5 be the probability that the challenger successfully ends and $b' = b$ in Game_5. As $R \in_R \mathbb{G}_1$, the equation $|\delta_5 - \frac{1}{2}| = 0$ holds.

With respect to the generation of C_b, from Game_4 to Game_5, the only modification is that $\hat{e}(\mathsf{H}_1(ID_E||t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T||t_{int}^*), C_2^*)^{SK_T}$ has been replaced with R, where $R \in_R \mathbb{G}_1$. As a result, Game_5 is identical to Game_4 unless $\hat{e}(\mathsf{H}_1(ID_E||t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T||t_{int}^*), C_2^*)^{SK_T}$ has been queried to H_2. Note that $SK_E^{(0)}$ is not required in answering the oracle queries. We immediately obtain $|\delta_5 - \delta_4| \le \epsilon_5$ which is negligible based on the BDH assumption.

In summary, we have $|\delta_0 - \delta_5| = \epsilon \le \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + \epsilon_5$. which are negligible. As a result, ϵ is negligible, and the lemma now follows. □

Lemma 5.3. *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.*

Proof sketch. Suppose an adversary $\mathcal{A}$ has the advantage ϵ in the attack game depicted in Figure 4.

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.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^* \in_R \mathcal{X},\ C_1^* = g^{r_1^*},$$

$$C_2^* = g^{r_2^*},\ C_3^* = \mathsf{H}_3(C_1^*||C_2^*)^{r_1^*},$$

$$C_4^* = M_b \oplus \mathsf{H}_2(\hat{e}(\mathsf{H}_1(ID_E||t_{eph_i}), C_1^*)^{SK_E^{(0)}} \cdot \hat{e}(\mathsf{H}_1(ID_T||t_{int}^*), C_2^*)^{SK_T}),$$

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

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

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 $\mathsf{Decrypt}$ query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$, the challenger answers as the following.

1. In step 4 of the game, if $C = C_b$, the challenger returns C', where

$$M' \in_R \{0,1\}^n,\ C_1' = C_1^*,\ C_2' = C_2^*,\ C_3' = C_3^*,$$

$$C_4' = M' \oplus C_4^*,\ Y \in_R \mathcal{Y},$$

$$C_5' = \mathsf{Encrypt}_1(Y, PK_{t_{eph_j}}^{(1)}), C_6' = \mathsf{H}_4(Y) \oplus (C_1'||C_2'||C_3'||C_4'),$$

$$C_7' = \mathsf{H}_2(Y||C_1'||C_2'||C_3'||C_4'||C_5'||C_6'),\ C' = (C_5', C_6', C_7').$$

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_2 such that

$$C_5 = \mathsf{Encrypt}_2(\tilde{X}, PK_U),\ C_6 = \tilde{C}_6, \quad (17)$$

$$\mathsf{H}_5(\tilde{X}) \oplus C_6 = \tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4, \text{ and } C_7 = \mathsf{H}_2(\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6). \quad (18)$$

If the input exists, the challenger returns C', where

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

$$C_4' = M' \oplus \tilde{C}_4,\ C_5' = \mathsf{Encrypt}_1(Y, PK_{t_{eph_j}}^{(1)}),$$

$$C_6' = \mathsf{H}_4(Y) \oplus (C_1'||C_2'||C_3'||C_4'),$$

$$C_7' = \mathsf{H}_2(Y||C_1'||C_2'||C_3'||C_4'||C_5'||C_6'),\ C' = (C_5', C_6', C_7').$$

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 $\mathsf{Decrypt}$ oracle queries.

- In the second case, there is a query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$ such that an oracle query to H_2 with the input $\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6$ (these values are determined by the equalities (17) and (18)) returns C_7, while the C_7 is chosen before the oracle query is made. Or,
- In the second case, there is a query with the input $(C = (C_5, C_6, C_7), t_{int}, t_{eph_j})$ such that oracle queries to H_2 with different inputs $\tilde{X}||\tilde{C}_1||\tilde{C}_2||\tilde{C}_3||\tilde{C}_4||\tilde{C}_5||\tilde{C}_6$ return C_7.

As H_2 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 $|\delta_1 - \delta_0| \le \epsilon_1 = \Pr[Evn]$ 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.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^*, X^\dagger \in_R \mathcal{X},$$

$$C_1^* = g^{r_1^*},\ C_2^* = g^{r_2^*},\ C_3^* = \mathsf{H}_3(C_1^*||C_2^*)^{r_1^*},$$

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

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

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

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_2 with $0 \| X^\dagger \| * \| * \| * \| * \| * \| *$. As $\mathcal{E}_2$ is one-way and $\mathsf{H}_5, \mathsf{H}_2$ 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 $|\delta_2 - \delta_1| \le \epsilon_2 = \Pr[Evn]$ 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.

$$r_1^*, r_2^* \in_R \mathbb{Z}_p,\ X^*, X^\dagger \in_R \mathcal{X},\ R \in \mathbb{G}_1,\ C_1^* = g^{r_1^*},$$

$$C_2^* = g^{r_2^*},\ C_3^* = \mathsf{H}_3(C_1^* \| C_2^*)^{r_1^*},\ C_4^* = M_b \oplus \mathsf{H}_2(R),$$

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

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

Regardless of the change, the game Game_3 is identical to Game_2 as $X^* \in_R \mathcal{X}$ and $\mathsf{H}_5, \mathsf{H}_2$ 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 $|\delta_0 - \delta_3| = \epsilon \le \epsilon_1 + \epsilon_2$. which are negligible. As a result, ϵ is negligible, and the lemma now follows. □

6. FURTHER REMARKS

In this section, we first compare the concepts of Ephemerizer (and Timed-Ephemerizer) with that of *Vanish* [8]. We then present an application of Timed-Ephemerizer in an outsourcing scenario. As Ephemerizer protocols (instead of Timed-Ephemerizer protocols) may be of independent interest, we provide a formalization for Ephemerizer.

6.1. A Comparison to *Vanish*

Geambasu *et al.* [8] claim that *Vanish* is superior to Ephemerizer (and Timed-Ephemerizer) in the sense that the latter requires the Ephemerizer to be trusted for securely delete the expired ephemeral private keys. In contrast, with *Vanish*, the user needs to trust the P2P network in the sense that a certain proportion of the nodes would not be seized by the adversary. This trust assumption seems valid, however, there is no guarantee. Recently, Wolchok *et al.* [15] have shown that Vanish is vulnerable to low-cost Sybil attacks. Besides, we have the following additional concerns.

With respect to P2P nodes, it is impossible to precisely determine the time period that a threshold subset of nodes stay in the network. Therefore, there is no precise guarantee on the expiration time of the sensitive data. There are two concerns here.

1. One is that the P2P nodes leave the network sooner than expected, which means that the sensitive data will become unrecoverable or unavailable when they are needed. This shortcoming has been noted in [8], and a statement is explicitly made that *Vanish* is suitable to protect sensitive data, for which the user cares more about privacy than its availability.
2. The other is that the P2P nodes leave the network later than expected, which means that the sensitive data remains recoverable or available when they should have been deleted. In this case, extra action should be taken to delete the private key shares and make the sensitive data unrecoverable.

In contrast, with Ephemerizer and Timed-Ephemerizer, precise guarantee on the expiration time of sensitive data is guaranteed, and the availability is guaranteed as long as the Ephemerizer (and the time server) are available.

With respect to the availability of data, the other concern is DoS and DDoS attacks. With *Vanish*, the user needs to retrieve the key shares from a threshold subset of P2P nodes in order to recover the symmetric key. In practice, a P2P network is easily subjected to DoS and DDoS attacks [16, 17], so is *Vanish*. Facing DoS and DDoS attacks, the availability of the protected data could be a problem for the underlying applications. For Ephemerizer (and Timed-ephemerizer), the Ephemerizer (and the time server) will be well protected because they are dedicated (commercial) entities for providing the services. The adversary may still try to mount DoS and DDoS attacks, but, the risks will be much less compared with that facing *Vanish*.

In summary, Ephemerizer (and Timed-Ephemerizer) can provide assured lifecycle where the guarantees are precise and provable from the perspective of security. In contrast, *Vanish* sacrifices availability for security, while the security guarantee relies on the underlying P2P network. Ephemerizer (and Timed-Ephemerizer) need dedicated third parties, namely the Ephemerizer and the time server, in order to provide the service. This additional infrastructure requirement may be regarded as a disadvantage, however, the complexity is paid back by the assured lifecycle and availability of sensitive data.

6.2. Application Example: Outsourcing Data Security

As an example, we consider the following data management problem in outsourcing activities.

Suppose Alice and Bob have signed an outsourcing contract, in which Alice wants to outsource part of her business to Bob. Suppose also that, during the outsourcing process, Bob needs to frequently access a large amount of Alice's private data. In this scenario, in order to protect the confidentiality of Alice's data, Alice and Bob need to deploy an efficient protocol which makes Alice's data available to Bob during the contract period and unrecoverable after the contract ends.

There exists two straightforward solutions, referred to as *On-demand transfer* and *Direct encryption*.

- *On-demand transfer*: This solution is that, Bob requests the data from Alice whenever needed and deletes them immediately afterwards. Clearly, this solution may introduce terrible communication complexity. In addition, for each request, Alice needs to authenticate Bob before granting the access.
- *Direct encryption*: This solution is asking Bob to encrypt and store Alice's data during the contract period and delete the data after the contract ends. However, clearly, this solution will put Alice's data in danger if Bob fails to securely delete the data after the contract ends.

With a Timed-Ephemerizer protocol, we have the following solution to solve the above problem.

1. Alice encrypts her data using a symmetric key K and runs the $\mathsf{Encrypt}$ algorithm to encapsulate K. Without loss of generality, suppose the ciphertexts are C_M and C_K respectively. Alice lets Bob store C_M and C_K.
2. When Bob needs to recover the data, he runs the Timed-Ephemerizer protocol, more specifically the $\mathsf{Decrypt}$ algorithm, to retrieve the symmetric key K. With K, Bob can recovers the data from C_M.

If we assume the plaintext data will only reside in Bob's volatile storage devices, Alice's data will be unrecoverable after the underlying ephemeral key pair has expired and been securely deleted by the Ephemerizer. In addition, Alice can set an initial disclosure time for her data. In practice, the Ephemerizer and the time server can be dedicated (and commercial) organizations (like Verisign being as a CA) that serve many users. Compared with the *On-demand transfer* solution, the current solution requires lower communication complexity since only a symmetric key needs to be retrieved; while, compared with *Direct encryption* solution, the current solution provides stronger guarantee that expired data will be unrecoverable.

It is worth noting that the above discussion assumes that Bob is honest and wishes to protect Alice's data. The solution implies that Bob does not need to revoke his key in order to stop access Alice's data. If bob is careless, then he might mistakenly store Alice's data in persistent storage devices. To prevent this, in practice, trusted software/hardware can be employed to guarantee that plaintext data never leaves volatile memory. However, if Bob is dishonest then he can always leak the information. To counter this kind of attack, pure technical measures, such as trusted software/hardware and Timed-Ephemerizer, may be insufficient, and other measures such as strict non-disclosure business agreement should be employed. Since it is out the scope of this paper, we omit the detailed description here.

6.3. Security Model for Ephemerizer

Formally, an Ephemerizer protocol involves the two types of entities: users and an Ephemerizer, and consists of the following polynomial-time algorithms.

- Setup'_E and Setup'_U: they are identical to Setup_E and Setup_U for Timed-Ephemerizer, respectively.
- $\mathsf{Encrypt}'(M, PK_U, PK_{t_{eph_j}})$: This algorithm outputs a ciphertext C. For the message M, t_{eph_j} is the expiration time. We explicitly assume that both t_{eph_j} and C should be sent to the user.
- $\mathsf{Decrypt}'(C, SK_U; SK_{t_{eph_j}})$: Interactively run between a user and the Ephemerizer, this algorithm outputs a plaintext M or an error symbol for the user.

With respect to Ephemerizer protocols, we distinguish the following two types of adversaries.

- Outsider security: This type of adversary wants to access data after its expiration time. It represents a malicious outside entity which has compromised the Ephemerizer and the user after the expiration time of the data.
- Insider security: This type of adversary represents a malicious Ephemerizer.

Definition 6.1. *An Ephemerizer protocol achieves outsider semantic security if any polynomial time adversary has only a negligible advantage in the following semantic security game (as shown in Figure 5), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

```
1. $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $1 \le j \le \mathcal{N} \overset{\$}{\leftarrow} \mathsf{Setup}'_E(\ell)$;
   $(PK_U, SK_U) \overset{\$}{\leftarrow} \mathsf{Setup}'_U(\ell)$
2. $(M_0, M_1, PK_{t_{eph_i}}) \overset{\$}{\leftarrow} \mathcal{A}^{(\mathsf{Decrypt}')}(SK_{t_{eph_j}}$ for $i < j \le \mathcal{N}, SK_U, PK_*)$
3. $b \overset{\$}{\leftarrow} \{0,1\}; C_b \overset{\$}{\leftarrow} \mathsf{Encrypt}'(M_b, PK_U, PK_{t_{eph_i}})$
4. $b' \overset{\$}{\leftarrow} \mathcal{A}^{(\mathsf{Decrypt}')}(C_b, SK_{t_{eph_j}}$ for $i < j \le \mathcal{N}, SK_U, PK_*)$
```

FIGURE 5. Semantic Security against Outsider Adversary

In the attack game, PK_* means all available public keys. 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 user.

1. The challenger runs Setup_E to generate $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $1 \le j \le \mathcal{N}$, and runs Setup_U to generate (PK_U, SK_U). All public parameters are given to the adversary.
2. The adversary can adaptively issue the following two types of $\mathsf{Decrypt}$ oracle queries.
   (a) D-type $\mathsf{Decrypt}$ oracle query: In each oracle query, the adversary impersonates the Ephemerizer and provides t_{eph_j} and C to the challenger, which then uses (C, SK_U) as input and runs the $\mathsf{Decrypt}$ algorithm with the adversary to decrypt C by assuming that the expiration time is t_{eph_j}.
   (b) E-type $\mathsf{Decrypt}$ query: In each oracle query, the adversary impersonates a user to the Ephemerizer and sends t_{eph_j} to the challenger, which uses $SK_{t_{eph_j}}$ as the input and runs the $\mathsf{Decrypt}$ 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 a timestamp t_{eph_i}. In this phase, the adversary can query for SK_U and $SK_{t_{eph_j}}$ for any $i < j \le \mathcal{N}$ with the following restriction: if SK_U has been queried, then any E-type $\mathsf{Decrypt}$ oracle query with the input t_{eph_j} for any $1 \le j \le i$ is forbidden.
3. The challenger picks a random bit $b \in \{0, 1\}$ and gives the adversary C_b as the challenge, where

$$C_b = \mathsf{Encrypt}'(M_b, PK_U, PK_{t_{eph_i}}).$$

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 an outsider one because it has access to the private keys SK_U and $SK_{t_{eph_j}}$ for any $i < j \le \mathcal{N}$.

Definition 6.2. *An Ephemerizer protocol achieves insider semantic security if any polynomial time adversary has only a negligible advantage in the following semantic security game (as shown in Figure 6), where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

In the attack game, PK_* means all available public keys. 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 user.

1. The adversary $\mathcal{A}$ generates $(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $1 \le j \le \mathcal{N}$. Note that the adversary may not follow the

1.	$(PK_{t_{eph_j}}, SK_{t_{eph_j}})$ for $1 \le j \le \mathcal{N} \overset{\$}{\leftarrow} \mathcal{A}(\ell)$; $(PK_U, SK_U) \overset{\$}{\leftarrow} \mathsf{Setup}'_U(\ell)$
2.	$(M_0, M_1, PK_{t_{eph_i}}) \overset{\$}{\leftarrow} \mathcal{A}^{(\mathsf{Decrypt}')}(SK_{t_{eph_j}}$ for $1 \le j \le \mathcal{N}, PK_*)$
3.	$b \overset{\$}{\leftarrow} \{0, 1\}; C_b \overset{\$}{\leftarrow} \mathsf{Encrypt}'(M_b, PK_U, PK_{t_{eph_i}})$
4.	$b' \overset{\$}{\leftarrow} \mathcal{A}^{(\mathsf{Decrypt}')}(C_b, SK_{t_{eph_j}}$ for $1 \le j \le \mathcal{N}, PK_*)$

FIGURE 6. Semantic Security against Insider Adversary

protocol specification. The challenger runs Setup_U to generate (PK_U, SK_U). The public key PK_U given to the adversary.

2. The adversary can adaptively issue the D-type $\mathsf{Decrypt}$ 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 a timestamp t_{eph_i}.
3. The challenger picks a random bit $b \in \{0, 1\}$ and gives the adversary C_b as the challenge, where

$$C_b = \mathsf{Encrypt}(M_b, PK_U, PK_{t_{eph_i}}).$$

4. The adversary can continue to query the $\mathsf{Decrypt}$ oracle as in Step 2.
5. The adversary $\mathcal{A}$ outputs b'.

In the above attack game, the adversary is an insider because it has access to all the private keys $SK_{t_{eph_j}}$ for any $1 \le j \le \mathcal{N}$.

7. RELATED WORK

In this section, we first briefly review some relevant works that focus on securely deleting expired sensitive data. We then briefly review the concept of Timed-Release Encryption.

7.1. Ephemerizer and Similar Protocols

Perlman [3, 4] proposed two Ephemerizer protocols without providing rigorous security proofs. One protocol uses a blind decryption technique. 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 user, the public key of the Ephemerizer, and the public key of the user. Nair *et al.* [5] has shown that the second protocol is vulnerable to attacks. In addition, Nair *et al.* [5] observed 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.* proposed an Ephemerizer protocol using identity-based public-key encryption [9, 18]. However, they have not provided any security analysis in a formal security model. In Section 3 we show that both the first protocol by Perlman and the protocol by Nair *et al.* are vulnerable to attacks.

Forward-secure encryption, e.g. [19], guarantees that if an attacker learns the state of the users cryptographic keys at some point in time, they should not be able to decrypt data encrypted at an earlier time. The security of a forward-secure encryption scheme relies on the requirement that the user will securely delete the expired private keys including backups. Ephemerizer relaxes this requirement on the user by shifting the key revocation responsibility to the Ephemerizer. Informally, Ephemerizer can be regarded as a three-party version of forward-secure encryption.

Geambasu *et al.* [8] introduced the concept of *Vanish* for the purpose of the self-destruction of sensitive data, which utilizes the dynamic nature of P2P networks where peer nodes dynamically join and leave the network. With *Vanish*, sensitive data is encrypted using a symmetric key, which is then divided into a number of shares using Shamir's secret sharing technique [18]. The key shares are distributed into a set of nodes (randomly chosen) in a P2P network, and the symmetric key becomes unrecoverable when a subset of a P2P nodes leave the network. Compared with Ephemerizer and Timed-Ephemerizer, *Vanish* cannot provide a precisely defined expiration time. In Section 6.1, we provide a detailed comparison between them. After [8], Wolchok *et al.* [15] have shown that Vanish is vulnerable to low-cost Sybil attacks. Recently, Wang, Yue, and Liu [20] proposed a solution similar to [8]. Inherently,it is also vulnerable to the attacks by Wolchok *et al.*.

7.2. 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 [21]. 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 [12]. Hwang, Yum, and Lee [22] 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 [23] 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 [12], 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. [24, 12]). 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.

8. CONCLUSION

In this paper we revisited the concept of Ephemerizer, proposed by Perlman, and show that some existing ephemerizer protocols possess vulnerabilities. We then 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.

1. With respect to public key encryption schemes [25], there are generally three levels of security guarantees (listed from the weakest to the strongest), namely one-wayness, IND-CPA (indistinguishability against chosen plaintext attacks), and IND-CCA (indistinguishability against chosen ciphertext attacks). In our security model, we have mainly focused on the chosen plaintext security, although some types of adversaries can interact with the challenger to obtain some information about the plaintext data. It is an interesting work to work out a security model and a new protocol to achieve the strongest security, namely considering chosen ciphertext attacks.
2. Another 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.
3. Another interesting research question is to use Timed-Ephemerizer as a tool to solve practical security problems. Note that, as an application of Ephemerizer, Perlman [26] proposes a file system that supports high availability of data with assured deletion.
4. Timed-Ephemerizer and most similar solutions rely on third parties to fulfill their functionalities. There is a practical concern that if the third parties refuse to cooperate then the data will be breached or lost. How to address this issue is yet another interesting question.

REFERENCES

[1] (2006) *National Industrial Security Program Operating Manual (NISPOM)*. DoD 5220.22-M.

[2] Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J. A., Feldman, A. J., Appelbaum, J., and Felten, E. W. (2008) Lest We Remember: Cold Boot Attacks on Encryption Keys. In van Oorschot, P. C. (ed.), *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pp. 45–60. USENIX Association.

[3] Perlman, R. (2005) The Ephemerizer: Making Data Disappear. *Journal of Information System Security*, **1**, 51–68.

[4] Perlman, R. (2005) The Ephemerizer: Making Data Disappear. Technical Report TR-2005-140. Sun Microsystems, Inc.

[5] Nair, S. K., Dashti, M. T., Crispo, B., and Tanenbaum, A. S. (2007) A Hybrid PKI-IBC Based Ephemerizer System. In Venter, H. S., Eloff, M. M., Labuschagne, L., Eloff, J. H. P., and von Solms, R. (eds.), *Proceedings of the IFIP TC-11 22nd International Information Security Conference (SEC 2007), 14-16 May 2007, Sandton, South Africa*, IFIP, **232**, pp. 241–252. Springer.

[6] Kannan, J., Altekar, G., Maniatis, P., and Chun, B. (2011) Making programs forget: Enforcing lifetime for sensitive data. *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems, May 9-11, Napa, CA, USA*, pp. 23–23. USENIX Association, Berkeley, CA, USA.

[7] Tang, Q. (2009) Timed-ephemerizer: Make assured data appear and disappear. *Public Key Infrastructures, Services and Applications - 6th European Workshop, EuroPKI 2009, Pisa, Italy, September 10-11, 2009*, Lecture Notes in Computer Science, **6391**, pp. 195–208. Springer.

[8] Geambasu, R., Kohno, T., Levy, A., and Levy, H. M. (2009) Vanish: Increasing Data Privacy with Self-Destructing Data. *18th USENIX Security Symposium, Montreal, Canada, August 10-14, 2009, Proceedings*.

[9] Boneh, D. and Franklin, M. K. (2001) Identity-based encryption from the weil pairing. In Kilian, J. (ed.), *21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, Lecture Notes in Computer Science, **2139**, pp. 213–229. Springer.

[10] Bellare, M. and Palacio, A. (2004) The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Franklin, M. K. (ed.), *24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, LNCS, **3152**, pp. 273–289. Springer.

[11] Damgård, I. (1991) Towards practical public key systems secure against chosen ciphertext attacks. In Feigenbaum, J. (ed.), *11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, LNCS, **576**, pp. 445–456. Springer.

[12] Rivest, R. L., Shamir, A., and Wagner, D. A. (1996) Time-lock puzzles and timed-release crypto. Technical Report Tech. Report MIT/LCS/TR-684. MIT LCS.

[13] Shamir, A. (1985) Identity-based cryptosystems and signature schemes. *Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, Lecture Notes in Computer Science, **196**, pp. 47–53. Springer.

[14] Shoup, V. (2006). Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/.

[15] Wolchok, S., Hofmann, O., N. Heninger, E. F., Halderman, A., Rossbach, C., Waters, B., and Witchel, E. (2010) Defeating vanish with low-cost sybil attacks against large dhts. *Proceedings of the Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, USA, 28th February - 3rd March 2010*. The Internet Society.

[16] Naoumov, N. and Ross, K. (2006) Exploiting p2p systems for ddos attacks. *Proceedings of the 1st International Conference on Scalable Information Systems, Infoscale 2006, Hong Kong, May 30-June 1, 2006* 47. ACM.

[17] Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., and Yu, H. (2005) OpenDHT: a public DHT service and its uses. *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Philadelphia, Pennsylvania, USA, August 22-26, 2005*, pp. 73–84. ACM.

[18] Shamir, A. (1979) How to share a secret. *Commun. ACM*, **22**, 612–613.

[19] Bellare, M. and Yee, B. (2003) Forward-security in private-key cryptography. *The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, pp. 1–18.

[20] Wang, G., Yue, F., and Liu, Q. (2013) A secure self-destructing scheme for electronic data. *J. Comput. Syst. Sci.*, **79**, 279–290.

[21] May, T. C. (1993) *Time-release crypto*.

[22] Hwang, Y., Yum, D., and Lee, P. (2005) Timed-release encryption with pre-open capability and its application to certified e-mail system. In Zhou, J., Lopez, J., Deng, R., and Bao, F. (eds.), *Information Security, 8th International Conference, ISC 2005, Singapore, September 20-23, 2005, Proceedings*, Lecture Notes in Computer Science, **3650**, pp. 344–358. Springer.

[23] Dent, A. W. and Tang, Q. (2007) Revisiting the security model for timed-release encryption with pre-open capability. In Garay, J. A., Lenstra, A. K., Mambo, M., and Peralta, R. (eds.), *Information Security, 10th International Conference, ISC 2007, Valparaíso, Chile, October 9-12, 2007, Proceedings*, Lecture Notes in Computer Science, **4779**, pp. 158–174. Springer.

[24] Cathalo, J., Libert, B., and Quisquater, J.-J. (2005) Efficient and non-interactive timed-release encryption. In Qing, S., Mao, W., Lopez, J., and Wang, G. (eds.), *Information and Communications Security, 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005, Proceedings*, Lecture Notes in Computer Science, **3783**, pp. 291–303. Springer-Verlag.

[25] Bellare, M., Desai, A., Pointcheval, D., and Rogaway, P. (1998) Relations among notions of security for public-key encryption schemes. In Krawczyk, H. (ed.), *18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, Lecture Notes in Computer Science, **1462**, pp. 26–45. Springer.

[26] Perlman, R. (2005) File system design with assured delete. *3rd International IEEE Security in Storage Workshop (SISW 2005), December 13, 2005, San Francisco, California, USA*, pp. 83–88. IEEE Computer Society, Washington, DC, USA.