Time Capsule Signature: Efficient and Provably Secure Constructions *

Bessie C. Hu[1], Duncan S. Wong[1], Qiong Huang[2], Guomin Yang[1], and Xiaotie Deng[1]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
{bessiehu,duncan,csyanggm,deng}@cs.cityu.edu.hk
[2] csqhuang@cityu.edu.hk

Abstract. Time Capsule Signature, first formalized by Dodis and Yum in Financial Cryptography 2005, is a digital signature scheme which allows a signature to bear a (future) time t so that the signature will only be valid at time t or later, when a trusted third party called time server releases time-dependent information for checking the validity of a time capsule signature. Also, the actual signer of a time capsule signature has the privilege to make the signature valid *before* time t.
In this paper, we provide a new security model of time capsule signature such that time server is not required to be fully trusted. Moreover, we provide two efficient constructions in random oracle model and standard model. Our improved security model and proven secure constructions have the potential to build some new E-Commerce applications.

Keywords: Time Capsule Signature

1 Introduction

Modern business is in nature the business for future. A contract signed now is a commitment for some future cooperation; a ticket bought now presents an entry permit at a specific time in the future; an option obtained now, in the derivative markets, ensures the privilege of buying/selling a stock at some time in the future. The success of these practices requires the integrity of credential releasers, and the involvement of an authority who can judge the rules for legal players. To realize these activities in E-Commerce platforms, a new primitive, which has a great promise to be a very useful tool, is called Time Capsule Signature [13].

A time capsule signature involves a signer (known as credential releaser), a verifier (known as credential receiver) and a time server (known as authority). The signer can issue a *future* signature indicated by some time information,

* The second author was supported by a grant from City University of Hong Kong (Project No. 7002001).

say t, and enjoys the following properties: 1) The credential receiver can verify immediately that a signature will become valid at time t. 2) The signature will automatically become valid at time t, even without the cooperation of signer. 3) The legal signer has the privilege to make the signature valid before time t.

Property 1 and 2 are easy to comprehend in the current practice. However, in a naive solution of signing a statement that 'the message m will become valid from time t', the verifier is required to be aware of the current time [13]. When time is generalized to arbitrary events, this becomes even more problematic. Moreover, signer has lost control of the validation time t once the statement is produced. For the variety of E-Commerce, we do need to provide signers the power to validate their future signature before the committed time t. For example, in the case of debt repayment, a borrower can sign a check to indicate the repayment day (e.g. due day), he may also have the desire to repay his debt earlier, so to improve his credit history. Of course he can sign another check indicating the actual repayment time, but the original check should be handled carefully to avoid 'double spending'. Time capsule signature supports this desirable feature with a process of making a signature valid at any time by the actual signer known as *prehatch*, as opposed to *hatch* the signature at time t when some additional information is published by the time server. We refer readers to [13] for more discussions on the applications of time capsule signature. Property 3 may also be captured in a signed statement that 'the signature of message m will become valid from time t, or when the signer release some secret information'. Again, such a statement has problems when time is generalized to arbitrary events.

The notion of time capsule signature was first formalized by Dodis and Yum [13] in 2005. Besides the above three properties, they also require that prehatched signature should be indistinguishable from hatched signature. For practical use of time capsule signature as discussed above, the indistinguishability between prehatched signature and hatched signature is actually undesirable. Since the purpose of prehatching is to make a signature valid before time t, the verifier can simply compare the time t with the current time to identify if a signature is prehatched or normally hatched. Furthermore, in some scenarios, we actually need to distinguish a prehatched signature from a hatched signature. In the above debt repayment case, a prehatched signature has to be identified for credit history checking. On the other hand, under the property of indistinguishability, the time server has to be fully trusted, otherwise, there is no way to tell if a signature which becomes valid before time t is generated by the actual signer or a cheating time server.

Therefore, in this paper, we remove the requirement of indistinguishability for time capsule signature while retaining all other properties. This allows us to modify the security model for capturing attacks launched by a cheating time server. Our generic construction is based on a new primitive called identity-based trapdoor relation (IDTR). We propose two efficient implementations for the IDTR primitive, one is proven secure in the random oracle model, the other in the standard model.

1.1 Related Work

The work on timed-release cryptography was first summarized and discussed by May [17] in 1993, and further work was carried out by Rivest et al. [20] in 1996. The main purpose of timed-release cryptography is to ensure that encryption, commitment or signature cannot be opened or valid until a predetermined future time. There are two main approaches for constructing such a scheme. The first approach, categorized as *time-lock puzzles* [20], is to design a computational problem which could be solved by continually computing for at least some required period of time. This approach is widely used in applications, like *verifiable time capsules* [2, 3], *timed commitments* [9], and some recently proposed systems [14, 15]. The tradeoff of this approach is that immense computational overhead has to be put on the receiver, that makes it impractical for most real-world applications.

The second approach relies on a trusted agent who releases time-dependent information exactly according to a pre-specified schedule. Previous work is mainly on timed-release encryption, which diversifies according to the involvement level of the trusted agent. May [17] suggested that the trusted agent should store messages until the time to release. Rivest et. al. [20] suggested that the agent should pre-compute pairs of public/private keys, publish the public keys first and then release the private keys one by one according to some pre-specified schedule. Most of the recent results [10, 5, 18, 11] are based on Boneh and Franklin's identity-based encryption (IBE) scheme [7]. In this paper, we also use Boneh-Franklin IBE as one of the implementations of our generic construction. In this implementation, we replace the identity in the IBE scheme with the claimed time t, but the technical details are different from previous constructions which are only for timed-release encryption. They will become clear when going through our construction in the subsequent sections of this paper.

Another stream of research based on trusted agents is optimistic fair exchange of digital signatures [1, 8, 12]. In those constructions, a trusted agent needs to resolve all signatures where the signers are refusing to validate the signatures. Scalability is the main issue of this approach. Recently, there is a new construction of time capsule signature [23] based on ring signature [19]. However, in their system, the time server needs to generate time-dependent information for each individual user, thus scalability is a main problem.

1.2 Paper Organization

In Sec. 2, we introduce some preliminaries. The definition and security model of time capsule signature is specified in Sec. 3. In Sec. 4, we define a new notion called Identity-based Trapdoor Relation (IDTR) and propose two concrete implementations which are proven to be secure in the random oracle model and the the standard model. In Sec. 5, we propose a generic construction of time capsule signature based on IDTR and analyze its security. In Sec. 6, we extend IDTR by adding a new property called Hiding, and use it to construct a distinguishable time capsule signature which could capture an attacker launched by a malicious time server. Finally, we conclude in Sec. 7.

2 Preliminaries

Identity Based Encryption. The notion of Identity Based Encryption (IBE) was introduced by Shamir in 1984 [21]. In such a mechanism, public key could be an arbitrary string, which is chosen from user's name, network address, etc; user private key is properly generated by a trusted third party (Key Generation Center), and the secret can be preserved as long as Key Generation Center does not release its master secret key. For IBE, a message can be encrypted for a receiver even before the corresponding private key is generated. To this extent, IBE is a good candidate of sending a message to the future.

The first practical IBE was proposed by Boneh and Franklin [7] in 2001. They proposed a basic IBE scheme, which is secure against chosen plaintext attack(IND-ID-CPA). By extending the basic scheme, a full scheme could be achieved with security against adaptive chosen ciphertext attack(IND-ID-CCA) in the random oracle model.

In 2005, Brent Waters [22] presented the first efficient Identity-Based Encryption scheme that is fully secure without random oracles. The proof of their scheme makes use of an algebraic method first used by Boneh and Boyen [6] and the security of the scheme is reduced to the decisional Bilinear Diffie-Hellman (BDH) assumption.

Based on IBE, in this paper, we propose a new notion called *Identity Based Trapdoor Relation* (IDTR) which can then be applied to the construction of time capsule signature scheme.

Computational Diffie-Hellman Assumption. Let $\mathbb{G}$ be a group of order p (p is a prime). The challenger chooses $a, b \in \mathbb{Z}_p$ at random and outputs $(g, A = g^a, B = g^b)$, where $g \in \mathbb{G}$. The adversary then attempts to output $g^{ab} \in \mathbb{G}$. An adversary $\mathcal{B}$ has at least ϵ advantage if

$$Pr[\mathcal{B}(g, g^a, g^b) = g^{ab}] \geq \epsilon$$

where the probability is taken over the random choices of a, b and the random bits consumed by $\mathcal{B}$.

Definition: The computational (t, ϵ)-DH assumption holds if no t-time adversary has at least ϵ advantage in the game above.

3 Time Capsule Signature

3.1 Definition

A *time capsule signature scheme* consists of eight PPT algorithms (TSSetup, UserSetup, TSig, TVer, TRelease, Hatch, PreHatch, Ver). The definition below follows that of Dodis and Yum [13].

1. TSSetup (Time Server Key Setup): On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a public/secret time release key pair (tpk, tsk).

2. UserSetup (User Key Setup): On input 1^k, it generates a user public/secret key pair (upk, usk).
3. TSig (Time Capsule Signature Generation): On input (m, usk, upk, t), where t is a time value from which the signature will become valid. It outputs a time capsule signature σ'_t.
4. TVer (Time Capsule Signature Verification): On input $(m, \sigma'_t, upk, tpk, t)$, it returns 1 (accept) or 0 (reject). A time capsule signature σ'_t is said to be valid if TVer returns 1 on it.
5. TRelease (Time Release): At the beginning of each time period T, $z_T \leftarrow \text{TRelease}(T, tsk)$ is published by the time server.
6. Hatch (Signature Hatch): On input $(m, \sigma'_t, upk, tpk, z_t)$, anyone can run this algorithm to get a hatched signature σ_t from a valid time capsule signature σ'_t.
7. PreHatch (Signature Prehatch): On input $(m, \sigma'_t, usk, tpk, t)$, the signer can run the algorithm to get a prehatched signature σ_t of a valid time capsule signature σ'_t before time t. However, if σ'_t is not valid, namely, $\mathsf{TVer}(m, \sigma'_t, upk, tpk, t) = 0$, then PreHatch should return $\perp$ which stands for unsuccessful prehatch.
8. Ver (Signature Verification): On input $(m, \sigma_t, upk, tpk, t)$, it returns 1 (accept) or 0 (reject).

Note that Time Server does not contact any user or need to know anything from any user.

3.2 Adversarial Model

There are three types of adversaries, $\mathcal{A}_I$, $\mathcal{A}_{II}$ and $\mathcal{A}_{III}$. $\mathcal{A}_I$ simulates a malicious signer whose aim is to produce a time capsule signature σ'_t, which looks good to a verifier, but cannot be hatched at time t. $\mathcal{A}_{II}$ simulates a malicious verifier who wants to hatch a time capsule signature before time t. $\mathcal{A}_{III}$ simulates a malicious time server who wants to forge a signature. Note that attacks launched by an outsider who wants to forge a signature can also be captured by $\mathcal{A}_{III}$. In the following, let $k \in \mathbb{N}$ be a security parameter.

Game I: Let $\mathcal{S}_I$ be the game simulator.

1. $\mathcal{S}_I$ executes $\mathsf{TSSetup}(1^k)$ to get (tpk, tsk).
2. $\mathcal{S}_I$ runs $\mathcal{A}_I$ on tpk. During the simulation, $\mathcal{A}_I$ can make queries onto TRelease.
3. $\mathcal{A}_I$ is to output $(m^*, t^*, \sigma'^*, upk)$.
4. $\mathcal{S}_I$ executes $\mathsf{TRelease}(t^*, tsk)$ to get z_t^*, and then executes $\mathsf{Hatch}(m^*, \sigma'^*, upk, tpk, z_t^*)$ to get σ^*.

$\mathcal{A}_I$ *wins* if $\mathsf{TVer}(m^*, \sigma'^*, upk, tpk, t) = 1$ and $\mathsf{Ver}(m^*, \sigma^*, upk, tpk, t) = 0$.

A time capsule signature scheme is secure in **Game I** if for every PPT algorithm $\mathcal{A}_I$, it is negligible for $\mathcal{A}_I$ to win the game. Note that we do not put any restriction on the generation of user public key upk. This is natural as in practice, $\mathcal{A}_I$ is normally the one who generates (upk, usk).

Game II: Let $\mathcal{S}_{II}$ be the game simulator.

1. $\mathcal{S}_{II}$ executes $\mathsf{TSSetup}(1^k)$ to get (tpk, tsk) and $\mathsf{UserSetup}(1^k)$ to get (upk, usk).
2. $\mathcal{S}_{II}$ runs $\mathcal{A}_{II}$ on tpk and upk. During the simulation, $\mathcal{A}_{II}$ can make queries onto TSig, $\mathsf{TRelease}$ and $\mathsf{PreHatch}$.
3. $\mathcal{A}_{II}$ is to output (m^*, t^*, σ^*).

$\mathcal{A}_{II}$ *wins* if $\mathsf{Ver}(m^*, \sigma^*, upk, tpk, t^*) = 1$, and $\mathcal{A}_{II}$ has never queried $\mathsf{TRelease}(t^*)$ and $\mathsf{PreHatch}(m^*, t^*, \cdot)$.

A time capsule signature scheme is secure in **Game II** if for every PPT algorithm $\mathcal{A}_{II}$, it is negligible for $\mathcal{A}_{II}$ to win the game.

Game III: Let $\mathcal{S}_{III}$ be the game simulator.

1. $\mathcal{S}_{III}$ executes $\mathsf{TSSetup}(1^k)$ to get (tpk, tsk), and $\mathsf{UserSetup}(1^k)$ to get (upk, usk).
2. $\mathcal{S}_{III}$ runs $\mathcal{A}_{III}$ on upk, tpk and tsk. During the simulation, $\mathcal{A}_{III}$ can make queries onto TSig, and $\mathsf{PreHatch}$.
3. $\mathcal{A}_{III}$ is to output (m^*, t^*, σ^*).

$\mathcal{A}_{III}$ *wins* if $\mathsf{Ver}(m^*, \sigma^*, upk, tpk, t^*) = 1$, and $\mathcal{A}_{III}$ has never queried $\mathsf{TSig}(m^*, \cdot)$ for time t^*.

A time capsule signature scheme is secure in **Game III** if for every PPT algorithm $\mathcal{A}_{III}$, it is negligible for $\mathcal{A}_{III}$ to win the game.

3.3 Discussion

One of the properties of time capsule signature in Dodis-Yum paper is ambiguity which ensures that a prehatched signature is indistinguishable with a hatched signature (with respect to the same message and time value t). Although this property may have independent interest, we notice that in common applications of time capsule signature described in Sec. 1 and in [13], this property is actually undesirable. Since the only purpose of prehatching a signature is to make the signature verifiable before time t. In this case, the verifier can simply check the time t against the current time for finding out if the signature is prehatched or normally hatched.

Our definition, instead, does not requires ambiguity. By this relaxation, we can construct more efficient time capsule signature schemes based on identity-based trapdoor relation (IDTR) in Sec. 4. We will see more discussions on this relaxation and explain that for some applications, it is actually important for the verifier to tell whether a signature is pre-hatched or hatched.

4 Identity-based Trapdoor Relation (IDTR)

A binary relation R is a subset of $\{0,1\}^* \times \{0,1\}^*$ and the language $\mathcal{L}_R$ is the set of α's for which there exist β such that $(\alpha, \beta) \in R$, i.e., $\mathcal{L}_R = \{\alpha | \exists \beta[(\alpha, \beta) \in R]\}$. We assume that (1) there is an efficient algorithm to decide whether $\alpha \in \mathcal{L}_R$ or

not, (2) if $(\alpha, \beta) \in R$, then the length of β is polynomially bounded in $|\alpha|$, and (3) there exists a short description D_R which specifies the relation R.

An *identity-based trapdoor relation* (IDTR) is a set of relations $\mathcal{R} = \{R_{id} | id \in I_{\mathcal{R}}\}$, where each relation R_{id} is called a trapdoor relation and there is a master trapdoor $mtd_{\mathcal{R}}$ for extracting the trapdoor td_{id} of each R_{id}. Formally, IDTR is specified by the following five probabilistic polynomial-time (PPT) algorithms (Gen, Sample, Extract, Invert, Check).

1. Gen : This algorithm is used to generate $\mathcal{R} = \{R_{id} | id \in I_{\mathcal{R}}\}$ where $I_{\mathcal{R}}$ is a finite set of indices. $\mathsf{Gen}(1^k)$ returns $D_{\mathcal{R}}$ (the description of $\mathcal{R}$) and $mtd_{\mathcal{R}}$ (the master trapdoor).
2. Sample : This sampling algorithm takes $(D_{\mathcal{R}}, id)$ as input and $\mathsf{Sample}_{D_{\mathcal{R}}}(id)$ returns a random commitment c and witness d such that $(c, d) \in R_{id}$.
3. Extract : This algorithm is used to extract the trapdoor of each relation by using $mtd_{\mathcal{R}}$. $\mathsf{Extract}_{mtd_{\mathcal{R}}}(id)$ returns the trapdoor $\mathsf{td}_{R_{id}}$ of relation R_{id}.
4. Invert : This algorithm is used to find a witness d for a given $c \in \mathcal{L}_{R_{id}}$ by using the trapdoor $\mathsf{td}_{R_{id}}$. If $c \in \mathcal{L}_{R_{id}}$, then $\mathsf{Invert}_{\mathsf{td}_{R_{id}}}(c)$ returns a witness $\hat{d}$ such that $(c, \hat{d}) \in R_{id}$.
5. Check : This algorithm is used to check the validity of a witness d on the commitment c. If $(c, d) \in R_{id}$, then $\mathsf{Check}_{D_{\mathcal{R}}, id}(c, d)$ returns 1 (accept). Otherwise, it returns 0 (reject).

Properties: One-wayness requires that no one is able to find the witness of a commitment if the trapdoor information is not given. Soundness requires that no one can produce a commitment whose witness cannot be found using Invert.

– One-wayness: Let O_{Extract} be an oracle simulating the trapdoor extraction procedure Extract and $Query(A, O_{\mathsf{Extract}})$ the set of queries an algorithm A asked to O_{Extract}. It states that the following probability is negligible for all PPT algorithm $A = (A_1, A_2)$:

$$\begin{aligned} &Pr[\mathsf{Check}_{D_{\mathcal{R}}, id^*}(c^*, \tilde{d}) = 1 \wedge id^* \notin Query(A, O_{\mathsf{Extract}})| \\ &(D_{\mathcal{R}}, mtd_{\mathcal{R}}) \leftarrow \mathsf{Gen}(1^k); (id^*, h) \leftarrow A_1^{O_{\mathsf{Extract}}}(D_{\mathcal{R}}); \\ &(c^*, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(id^*); \tilde{d} \leftarrow A_2^{O_{\mathsf{Extract}}}(id^*, c^*, h)] \end{aligned}$$

– Soundness: We require that the following probability should be negligible for all algorithm B:

$$\begin{aligned} &Pr[R_{id^*} \in \mathcal{R} \wedge c^* \in \mathcal{L}_{R_{id^*}} \wedge \mathsf{Check}_{D_{\mathcal{R}}, id^*}(c^*, \tilde{d}) = 0| \\ &(D_{\mathcal{R}}, mtd_{\mathcal{R}}) \leftarrow \mathsf{Gen}(1^k); (c^*, id^*) \leftarrow B(D_{\mathcal{R}}, mtd_{\mathcal{R}}); \\ &\mathsf{td}_{R_{id^*}} \leftarrow \mathsf{Extract}_{mtd_{\mathcal{R}}}(id^*); \tilde{d} \leftarrow \mathsf{Invert}_{\mathsf{td}_{R_{id^*}}}(c^*)] \end{aligned}$$

***Discussion*:** The definition of IDTR above is much like the definition of Dodis and Yum's Identity-based Hard-to-Invert Relation (ID-THIR) [13]. ID-THIR has an *ambiguity* property which requires that witness $\hat{d}$ inverted from c given $\mathsf{td}_{R_{id}}$ is computationally indistinguishable from d obtained from $\mathsf{Sample}_{D_{\mathcal{R}}}(id)$ for the

same commitment c. To facilitate our construction of time capsule signature under new definition in Sec. 3, we do not require ambiguity property in the definition of IDTR above. We will see that with this relaxation we can construct much more efficient schemes then that in [13].

4.1 Implementations of IDTR

In this section, we propose two concrete constructions of IDTR, one based on Boneh and Franklin's IBE whose security has been proven in the random oracle model [7], and the other one based on Waters' IBE whose security has been proven in the standard model [22].

Implementation 1: In Random Oracle Model. An IBE scheme consists of four PPT algorithms (Setup, KeyGen, Encrypt, Decrypt). The Boneh-Franklin scheme [7] is described as follows:

1. Setup : Given a security parameter $k \in \mathbb{N}$, generate a prime q, two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order q, and an admissible bilinear map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $|q|$ is some polynomial in k. Choose a random generator $P \in \mathbb{G}_1$, pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$. Choose a cryptographic hash function H_1: $\{0,1\}^* \rightarrow \mathbb{G}_1$, another hash function H_2: $\mathbb{G}_2 \rightarrow \{0,1\}^k$, and the security analysis will view H_1, H_2 as random oracles [4]. The message space is $\mathcal{M} = \{0,1\}^k$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^k$. The system public key is $mpk = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, k, P, P_{pub}, H_1, H_2 \rangle$. The master secret key msk is $s \in \mathbb{Z}_q^*$.
2. KeyGen : For a given string $id \in \{0,1\}^*$ the algorithm computes $Q_{id} = H_1(id) \in \mathbb{G}_1$, and sets the private key sk_{id} to be sQ_{id} where s is the master secret key.
3. Encrypt : To encrypt $m \in \mathcal{M}$ under the public key id, the algorithm computes $Q_{id} = H_1(id) \in \mathbb{G}_1$, chooses a random $r \in \mathbb{Z}_q^*$, and sets the ciphertext to be $c = \langle rP, m \oplus H_2(g_{id}^r) \rangle$ where $g_{id} = \hat{e}(Q_{id}, P_{pub}) \in \mathbb{G}_2$.
4. Decrypt : Given the private key $sk_{id} \in \mathbb{G}_1$, a ciphertext $c = \langle c_1, c_2 \rangle \in \mathcal{C}$ can be decrypted by computing $c_2 \oplus H_2(\hat{e}(sk_{id}, c_1)) = m$.

An IDTR based on the IBE above is constructed as follows:

1. Gen : Run $\mathsf{Setup}(1^k)$, and set $\mathcal{L}_R = \mathcal{C}$, $D_{\mathcal{R}} = mpk$, and $mtd_{\mathcal{R}} = msk$.
2. Sample : Given $D_{\mathcal{R}}$ and id, randomly pick $m \in \mathcal{M}$ and compute $c = \mathsf{Encrypt}_{id}(m)$. Let $r \in \mathbb{Z}_q^*$ be the randomness used in Encrypt. Set witness $d = \langle rQ_{id}, P_{pub}, m \rangle$.
3. Extract : Given a string $id \in \{0,1\}^*$, compute $sk_{id} = \mathsf{KenGen}_{mpk,msk}(id)$, and set $\mathsf{td}_{R_{id}} = sk_{id}$.
4. Invert : Given trapdoor $\mathsf{td}_{R_{id}} \in \mathbb{G}_1$ and a commitment $c = \langle c_1, c_2 \rangle \in \mathcal{C}$, compute $m = \mathsf{Decrypt}_{sk_{id}}(c)$, and set the witness $\hat{d} = \langle \mathsf{td}_{R_{id}}, c_1, m \rangle$.
5. Check : Given $D_{\mathcal{R}}$, id, $c = \langle c_1, c_2 \rangle \in \mathcal{C}$, $d = \langle d_1, d_2, d_3 \rangle$ (where $d_1, d_2 \in \mathbb{G}_1$, and $d_3 \in \mathcal{M}$) , if $d_2 = P_{pub}$, $\hat{e}(d_1, P) = \hat{e}(c_1, Q_{id})$, and $c_2 = d_3 \oplus H_2(\hat{e}(d_1, d_2))$, return 1. Else if $d_1 = \mathsf{td}_{R_{id}}$, $d_2 = c_1$, $\hat{e}(d_1, P) = \hat{e}(P_{pub}, Q_{id})$ and $c_2 = d_3 \oplus H_2(\hat{e}(d_1, d_2))$, return 1. Otherwise, return 0.

One-wayness. In the game of one-wayness, an adversary A has access to the Extract oracle of all id other than id^*. This oracle is simulated by performing KeyGen of the underlying IBE scheme. A wins if it can find secret key sk_{id^*} and plaintext m^*. However, the semantic security (IND-ID-CPA) [7] of the underlying IBE attains that any PPT adversary will have negligible advantage in distinguishing m^* with another m in $\mathcal{M}$. If A succeeds, it is easy to see that we can also distinguish m^*, which contradicts the security of the underlying IBE scheme.

Soundness. An adversary B wins if it can generate a value c^* which is not able to decrypt under sk_{id^*}. In the underlying IBE scheme, this will not be the case even when B knows msk. Given id^*, sk_{id^*} can always be properly generated with the knowledge of msk. As long as c^* is in the ciphertext domain, a valid plaintext m^* can always be retrieved.

Remark: This construction of IDTR in random oracle model based on the Boneh-Franklin IBE scheme is much more efficient than the OR-proof for ID-THIR [13].

Implementation 2: In Standard Model. We now review Waters' IBE [22] and propose a construction for IDTR based on this scheme.

1. Setup : Given a security parameter $k \in \mathbb{N}$, generate a prime p, two groups $\mathbb{G}_1, \mathbb{G}_2$ of order p, and an admissible bilinear map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $|p|$ is some polynomial in k. Choose a random generator $g \in \mathbb{G}_1$, pick a random $\alpha \in \mathbb{Z}_p^*$ and set $g_1 = g^\alpha$. Choose random values $g_2, u' \in \mathbb{G}_1$, and a random k-length vector $U = (u_i)$, whose elements are chosen uniformly at random from $\mathbb{G}_1$. The message space is $\mathcal{M} \subseteq \mathbb{G}_2$. The ciphertext space is $\mathcal{C} = \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1$. The system public key is $mpk = \langle p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, k, g, g_1, g_2, u', U \rangle$. The master secret key msk is g_2^α.
2. KeyGen : Let v be an k-bit string representing an identity id, v_i denote the ith bit of v, and $\mathcal{V} \subseteq 1, ..., k$ be the set of all i for which $v_i = 1$. ($\mathcal{V}$ is the set of indices for which the bitstring v is set to 1.) Randomly select $r \in \mathbb{Z}_p^*$ and construct the private key sk_{id} as:

$$sk_{id} = \langle g_2^\alpha (u' \prod_{i \in \mathcal{V}} u_i)^r, g^r \rangle$$

3. Encrypt : To encrypt $m \in \mathcal{M}$ for an identity v, randomly select $t \in \mathbb{Z}_p^*$ and construct the ciphertext c as:

$$c = \langle \hat{e}(g_1, g_2)^t m, g^t, (u' \prod_{i \in \mathcal{V}} u_i)^t \rangle$$

4. Decrypt : Given the private key $sk_{id} = \langle sk_1, sk_2 \rangle$, a ciphertext $c = \langle c_1, c_2, c_3 \rangle \in \mathcal{C}$ can be decrypted as:

$$c_1 \frac{\hat{e}(sk_2, c_3)}{\hat{e}(sk_1, c_2)} = (\hat{e}(g_1, g_2)^t m) \frac{\hat{e}(g^r, (u' \prod_{i \in \mathcal{V}} u_i)^t)}{\hat{e}(g_2^{\alpha} (u' \prod_{i \in \mathcal{V}} u_i)^r, g^t)}$$
$$= (\hat{e}(g_1, g_2)^t m) \frac{\hat{e}(g, (u' \prod_{i \in \mathcal{V}} u_i)^{rt})}{(\hat{e}(g_1, g_2)^t) \hat{e}((u' \prod_{i \in \mathcal{V}} u_i)^{rt}, g)} = m$$

Based on Waters' IBE: $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$; $sk_{id} = \mathsf{KenGen}_{mpk,msk}(id)$; $c = \mathsf{Encrypt}_{id}(m)$; $m = \mathsf{Decrypt}_{sk_{id}}(c)$, an IDTR can be constructed as follows:

1. Gen : Given $k \in \mathbb{N}$, execute $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$ and set $\mathcal{L}_R = \mathcal{C}$, $D_{\mathcal{R}} = mpk$, and $mtd_{\mathcal{R}} = msk$.
2. Sample : Given $D_{\mathcal{R}}$ and id, randomly pick $m \in \mathbb{G}_2$ and compute $c = \mathsf{Encrypt}_{id}(m)$. Let $t \in \mathbb{Z}_p^*$ be the randomness in producing c. Set witness d to
$$d = \langle g_2^t, g_1 (u' \prod_{i \in \mathcal{V}} u_i), g_2, c_3, m \rangle$$
3. $\mathsf{Extract}$: Let v be an k-bit identity id, compute $sk_{id} = \mathsf{KenGen}_{mpk,msk}(id)$ and set $\mathsf{td}_{R_{id}} = sk_{id}$.
4. Invert : Given trapdoor $\mathsf{td}_{R_{id}}$ and a commitment $c = \langle c_1, c_2, c_3 \rangle \in \mathcal{C}$, compute $m = \mathsf{Decrypt}_{sk_{id}}(c)$, and set the witness to
$$\hat{d} = \langle sk_1, c_2, sk_2, c_3, m \rangle$$
5. Check : Given $D_{\mathcal{R}}$, id, $c = \langle c_1, c_2, c_3 \rangle \in \mathcal{C}$, $d = \langle d_1, d_2, d_3, d_4, d_5 \rangle$ (where $d_1, d_2, d_3, d_4 \in \mathbb{G}_1$, and $d_5 \in \mathcal{M}$), if $d_2 = g_1 (u' \prod_{i \in \mathcal{V}} u_i)$, $d_3 = g_2$,$d_4 = c_3$, $\hat{e}(d_1, u' \prod_{i \in \mathcal{V}} u_i) = \hat{e}(g_2, c_3)$, and $c_1 = M \frac{\hat{e}(d_1, d_2)}{\hat{e}(d_3, d_4)}$, return 1(**). Else if $d_1 = sk_1$, $d_2 = c_2$, $d_3 = sk_2$, $d_4 = c_3$, $\hat{e}(sk_1, g) = \hat{e}(g_2, g_1) \cdot \hat{e}(u' \prod_{i \in \mathcal{V}} u_i, sk_2)$, and $c_1 = m \frac{\hat{e}(d_1, d_2)}{\hat{e}(d_3, d_4)}$, return 1. Otherwise, return 0.

(**):The check will pass because:

$$m \frac{\hat{e}(d_1, d_2)}{\hat{e}(d_3, d_4)} = m \frac{\hat{e}(g_2^t, g_1 (u' \prod_{i \in \mathcal{V}} u_i))}{\hat{e}(g_2, (u' \prod_{i \in \mathcal{V}} u_i)^t)}$$
$$= m \frac{\hat{e}(g_2^t, g_1) \cdot \hat{e}(g_2^t, (u' \prod_{i \in \mathcal{V}} u_i))}{\hat{e}(g_2, (u' \prod_{i \in \mathcal{V}} u_i)^t)} = m \cdot \hat{e}(g_2^t, g_1) = c_1$$

Similar to the first implementation, the proof of $\mathsf{One\text{-}wayness}$ can be reduced to IND-ID-CPA security of Waters' IBE scheme. $\mathsf{Soundness}$ also holds since a valid $c \in \mathcal{C}$ can always be decrypted to a message m for a given sk_{id}.

***Discussion*:** Note that given $c \in \mathcal{C}$, we only require that an adversary is not able to compute the entire m for a randomly chosen $m \in \mathbb{G}_2$. In other words, we do not need IND-ID-CPA [7] security. Although both of the constructions could achieve IND-ID-CPA, this is not a necessity in our security notion.

5 Generic Construction of Time Capsule Signature

We now describe our generic construction of time capsule signature scheme. Our construction is based on the identity-based trapdoor relation (IDTR) defined in Sec. 4.

Let (Set, Sig, Verify) be the key generation, signature generation and verification algorithms of an ordinary signature scheme, and (Gen, Sample, Extract, Invert, Check) be the tuples of IDTR.

1. TSSetup: Let $k \in \mathbb{N}$ be a security parameter. The Time Sever gets $(D_{\mathcal{R}}, mtd_{\mathcal{R}}) \leftarrow \mathsf{Gen}(1^k)$ and sets public/secret time release key pair $(tpk, tsk) = (D_{\mathcal{R}}, mtd_{\mathcal{R}})$.
2. UserSetup: Each user runs $(pk, sk) \leftarrow \mathsf{Set}(1^k)$ and sets $(upk, usk) = (pk, sk)$.
3. TSig: To generate a time capsule signature on a message m for a future time t, the signer gets a commitment/witness pair $(c, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(t)$, then computes $s \leftarrow \mathsf{Sig}_{usk}(m\|c\|t)$. The time capsule signature σ'_t is (s, c). The signer stores the witness d.
4. TVer: A verifier checks if $\sigma'_t{=}(s, c)$ is a valid time capsule signature by checking whether $c \in \mathcal{L}_{R_t}$ and s is a valid standard signature under public key upk, that is, check if $\mathsf{Verify}_{upk}(m\|c\|t, s) = 1$. If both are correct, output 1; otherwise, output 0.
5. TRelease: At the beginning of each time period T, the Time Server gets $td_{R_T} \leftarrow \mathsf{Extract}_{tsk}(T)$ and publishes td_{R_T} as z_T.
6. Hatch: To hatch a time capsule signature $\sigma'_t = (s, c)$, a party computes $\hat{d} \leftarrow \mathsf{Invert}_{td_{R_t}}(c)$. The hatched signature is $\sigma_t = (s, c, \hat{d})$.
7. PreHatch: To prehatch a valid time capsule signature $\sigma'_t{=}(s, c)$, the signer retrieves stored value d, and sets the prehatched signature to $\sigma_t = (s, c, d)$. However, if $\mathsf{TVer}(m, \sigma'_t, upk, tpk, t) = 0$, then the algorithm outputs $\perp$.
8. Ver: For a given prehatched (or hatched) signature $\sigma_t = (s, c, d)$, the verifier checks the validity of (c, d) by running $\mathsf{Check}_{tpk,t}(c, d)$. Then, it verifies s on $m\|c\|t$ by running $\mathsf{Verify}_{upk}(m\|c\|t, s)$. If both verifications are correct, output 1; otherwise, output 0.

5.1 Security Analysis

Theorem 1. *The proposed time capsule signature scheme is secure if the underlying public key signature scheme is existentially unforgeable against adaptive chosen message attacks (euf-cma) [16] and the IDTR has the properties of one-wayness and soundness.*

Proof. We prove the security of our proposed time capsule signature scheme against Game I, Game II and Game III.

Security Against Game I: $\mathcal{A}_I$ wins the game if he can generate a valid time capsule signature $\sigma'_t{=}(s, c)$ such that $c \in \mathcal{L}_{R_t}$, and $\mathsf{Ver}_{upk}(m||c||t, s) {=}1$. Moreover, no party can obtain a witness $\tilde{d} = \mathsf{Invert}_{td_{R_t}}(c)$ such that $\mathsf{Check}_{tpk,t}(c, \tilde{d})$ $= 1$, where $td_{R_t} \leftarrow \mathsf{Extract}_{tsk}(t)$ is released by the Time Server. This contradicts the Soundness property of IDTR. Thus, the proposed time capsule signature

scheme is secure against **Game I** if underlying IDTR satisfies the Soundness property.

Security Against Game II: We construct an adversary $\mathcal{B}$ which breaks the One-wayness of IDTR with non-negligible advantage if $\mathcal{A}_{II}$ forges a valid signature σ. Let (m^*, t^*, σ^*) be a successful forgery generated by $\mathcal{A}_{II}$. Since the underlying standard signature scheme (Set, Sig, Verify) is euf-cma, $\mathcal{A}_{II}$ has overwhelming probability to have obtained the corresponding time capsule signature σ'^* from oracle TSig rather than forging σ' on its own.

The game between the IDTR One-wayness challenger and adversary $\mathcal{B}$ starts when the challenger generates $D_{\mathcal{R}}$ and $mtd_{\mathcal{R}}$ by running $\mathsf{Gen}(1^k)$. After receiving $D_{\mathcal{R}}$ from the challenger, $\mathcal{B}$ interacts with $\mathcal{A}_{II}$ in Game II as follows:

$\mathcal{B}$ gets a random public/private key pair $(pk, sk) \leftarrow \mathsf{Set}(1^k)$, sets $(upk, usk) = (pk, sk)$, $tpk = D_{\mathcal{R}}$, and gives (tpk, upk) to $\mathcal{A}_{II}$.

$\mathcal{B}$ manages a list $L = \{(m_i, t_i, s_i, c_i, d_i)\}$ for answering $\mathcal{A}_{II}$'s queries on PreHatch. Let q_{TSig} be the total number of TSig queries made by $\mathcal{A}_{II}$ and r be the random number chosen by $\mathcal{B}$ in the interval of $[1, q_{\mathsf{TSig}}]$. $\mathcal{B}$ responds to the i-th TSig query (m_i, t_i) as follows:

- If $i = r$, $\mathcal{B}$ sends t_r to the IDTR One-wayness challenger and receives a random commitment $c \in R_{t_r}$ from the challenger. $\mathcal{B}$ sets $c_r = c$ and computes $s_r = \mathsf{Sig}_{usk}(m_r \| c_r \| t_r)$. $\mathcal{B}$ returns $\sigma'_{t_r} = (s_r, c_r)$ to $\mathcal{A}_{II}$ and stores $(m_r, t_r, s_r, c_r, \perp)$ in the list L.
- If $i \neq r$, $\mathcal{B}$ gets a random commitment/witness pair (c_i, d_i) generated from $\mathsf{Sample}_{D_{\mathcal{R}}}$ and computes $s_i = \mathsf{Sig}_{usk}(m_i \| c_i \| t_i)$. $\mathcal{B}$ returns $\sigma'_{t_i} = (s_i, c_i)$ to $\mathcal{A}_{II}$ and stores $(m_i, t_i, s_i, c_i, d_i)$ in L.

To simulate oracle TRelease, say on query t_i from $\mathcal{A}_{II}$, $\mathcal{B}$ relays t_i to the trapdoor extraction oracle Extract simulated by the IDTR One-wayness challenger and gets $td_{\mathcal{R}_{t_i}}$. If $t_i = t_r$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns $z_{t_i} = td_{\mathcal{R}_{t_i}}$ to $\mathcal{A}_{II}$.

To simulate oracle PreHatch, say on query (m_i, t_i, s_i, c_i), $\mathcal{B}$ checks whether the query is in the list L or not. If (m_i, t_i, s_i, c_i) is in the list L, and equal to (m_r, t_r, s_r, c_r), $\mathcal{B}$ aborts. If (m_i, t_i, s_i, c_i) is in the list L, and not equal to (m_r, t_r, s_r, c_r), $\mathcal{B}$ extracts d_i from L and gives a prehatched signature $\sigma_{t_i} = (s_i, c_i, d_i)$ to $\mathcal{A}_{II}$. If (m_i, t_i, s_i, c_i) is not in L, since $\mathcal{A}_{II}$ does not know usk and this case implies that s_i is not generated by $\mathcal{B}$ on $m_i \| c_i \| t_i$, due to the euf-cma assumption of the underlying standard signature, it is negligible to have s_i be valid. Hence this case will happen with negligible chance. Therefore, for this case, $\mathcal{B}$ returns $\perp$.

When $\mathcal{A}_{II}$ outputs the forgery (m^*, t^*, σ^*) where $\sigma^* = (s^*, c^*, d^*)$, $\mathcal{B}$ verifies whether the forgery passes the verification algorithm Ver, and $(m^*, t^*, s^*, c^*) = (m_r, t_r, s_r, c_r)$. If so, $\mathcal{B}$ outputs the witness d^*. Otherwise, it chooses a d_B randomly and outputs d_B. The probability that $\mathcal{B}$ does not abort during the simulation and has a right guess of r is at least $1/q_{\mathsf{TSig}}$ since r is randomly chosen. Therefore, if $\mathcal{A}_{II}$ forges with a probability ϵ, $\mathcal{B}$ succeeds in breaking the One-wayness of IDTR with probability $\epsilon \geq \epsilon / q_{\mathsf{TSig}}$.

Security Against Game III: To show the security against Game III, we convert any adversary $\mathcal{A}_{III}$ which wins in Game III to a forger $\mathcal{F}$ against the underlying standard signature scheme. $\mathcal{F}$ gets pk as an input, and has access to signing oracle Sig of the signature scheme as described in the euf-cma model [16]. $\mathcal{F}$ simulates Game III for $\mathcal{A}_{III}$ as follows:

$\mathcal{F}$ gets $(D_{\mathcal{R}}, mtd_{\mathcal{R}}) \leftarrow \mathsf{Gen}(1^k)$ and gives $(upk, tpk, tsk) = (pk, D_{\mathcal{R}}, mtd_{\mathcal{R}})$ to $\mathcal{A}_{III}$. $\mathcal{F}$ simulates TSig on query (m_i, t_i) by getting $(c_i, d_i) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(t_i)$ and obtaining $s_i \leftarrow \mathsf{Sig}(m_i \| c_i \| t_i)$ from signing oracle Sig. $\mathcal{F}$ stores (m_i, c_i, d_i, t_i) in a list $L = \{(m_i, c_i, d_i, t_i)\}$ for answering $\mathcal{A}_{III}$'s queries to $\mathsf{PreHatch}$. To simulate $\mathsf{PreHatch}$ on query (m_i, t_i, s_i, c_i), $\mathcal{F}$ verifies if s_i is a valid signature on $m_i \| c_i \| t_i$.

- If s_i is valid, $\mathcal{F}$ checks if (m_i, c_i, t_i) is in the list L. If so, $\mathcal{F}$ gives the corresponding d_i to $\mathcal{A}_{III}$. Otherwise, s_i is a new signature value and $\mathcal{F}$ succeeds in producing a new forgery s_i on $m_i \| c_i \| t_i$.
- If s_i is not valid, $\mathcal{F}$ returns $\perp$ due to the same reason as shown above in the Security Against Game II.

Finally, when $\mathcal{A}_{III}$ outputs a forgery (m^*, t^*, σ_t^*) where $\sigma_t^* = (s^*, c^*, d^*)$, $\mathcal{F}$ outputs a signature s^* on message $m^* \| c^* \| t^*$. Therefore, if $\mathcal{A}_{III}$ succeeds with a probability ϵ, $\mathcal{F}$ succeeds in producing a new forgery with at least probability ϵ. □

6 Distinguishable Time Capsule Signature

As discussed in Sec 3.3, the ambiguity between a prehatched signature and a hatched signature may not be desirable in practice. Moreover, in some scenarios, there are demands to distinguish a prehatched signature from a hatched signature. In the case of debt repayment, as an example, if a borrower repays his debt before the actual due date, he can improve his credit history or get extra reward. Then the signature for validating the payment check should be determined on whether it is prehatched or hatched.

Our generic construction of time capsule signature can be extended to capture the need of distinguishability. In the following, we first extend the IDTR (identity-based trapdoor relation). We then modify our construction based on the extended IDTR.

6.1 Extended IDTR

The extended IDTR (identity-based trapdoor relation) has seven PPT algorithms associated (Gen, Sample, Reveal, $\mathsf{Extract}$, Invert, CheckS, CheckI). The settings of Gen, Sample, $\mathsf{Extract}$, and Invert remain the same as in IDTR. Reveal is used to print out a 'sampled' witness. Check in IDTR is replaced by two separated functions CheckS and CheckI, which are used to check the validity of sampled witness and inverted witness, respectively.

- Reveal: Given $c \in \mathcal{L}_{R_{id}}$, if there is a pair (c, d) in a sampling list defined by $List = \{(c, d, id)\}$ where $(c, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(id)$, $\mathsf{Reveal}_{id}(c)$ returns witness d. Otherwise, it returns $\perp$.
- CheckS: For any $(c, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(id)$, we have $\mathsf{CheckS}_{D_{\mathcal{R}},id}(c, d)$ return 1 (accept); otherwise, it returns 0 (reject).
- CheckI: Given $(c, \hat{d}) \in R_{id}$, where $\hat{d} \leftarrow \mathsf{Invert}_{td_{\mathcal{R}_{id}}}(c)$, $\mathsf{CheckI}_{D_{\mathcal{R}},id}(c, \hat{d})$ returns 1 (accept). Otherwise, it returns 0 (reject).

With this modification, the extended IDTR can be used to achieve another property called Hiding, which is beyond One-wayness and Soundness. Hiding captures a malicious system master (e.g. a malicious Time Server) who aims to forge a sampled witness for a given commitment.

- Hiding: Let O_{Sample} and O_{Reveal} be oracles simulating the procedures of Sample and Reveal, respectively, where O_{Sample} only returns a commitment for each query. Let $Query(A, O_{\mathsf{X}})$ be the set of queries an algorithm A asked to O_{X}, where X can be Sample or Reveal. Note that A can only obtain commitment c from O_{Sample}. It states that the following probability is negligible for all PPT algorithm A:

$$\begin{aligned} Pr[\mathsf{CheckS}_{D_{\mathcal{R}},id^*}(c^*, d^*) = 1 \wedge c^* \in Query(A, O_{\mathsf{Sample}}) \\ \wedge\ c^* \notin Query(A, O_{\mathsf{Reveal}}) | (D_{\mathcal{R}}, mtd_{\mathcal{R}}) \leftarrow \mathsf{Gen}(1^k); \\ (c^*, d^*, id^*) \leftarrow A^{O_{\mathsf{Sample}} O_{\mathsf{Reveal}}}(D_{\mathcal{R}}, mtd_{\mathcal{R}})] \end{aligned}$$

For One-wayness and Soundness, we refer readers to Sec. 4 for their definitions while replacing Check in One-wayness with CheckS and CheckI, and replacing Check in Soundness with CheckI.

6.2 A Generic Construction of Extended IDTR

Let $\mathcal{E}$ be an IBE scheme. Let $\mathcal{E}.\mathsf{Enc}(mpk, id, m; r)$ be $\mathcal{E}$'s encryption algorithm which encrypts message m under identity id and master public key mpk using randomness r. We say that $\mathcal{E}$ is injective if it satisfies the following condition:

Injective: For every master public key mpk and every identity id, for every ciphertext e of a message m under mpk and id, there exists *at most one* randomness r such that $e = \mathcal{E}.\mathsf{Enc}(mpk, id, m; r)$.

In the literature, many IBE schemes are injective, like BasicIdent and FullIdent proposed by Boneh and Franklin [7], and Waters' IBE [22].

Suppose $\mathcal{E} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ is an injective encryption scheme with IND-ID-CPA security [7], $\mathcal{MSP}$ is the message space, and $\mathcal{RSP}$ is the space of randomness used in $\mathcal{E}.\mathsf{Enc}$. Let $f : \{0,1\}^{\ell(k)} \rightarrow \mathcal{RSP}$ be a one-way function (or a hash function). We now give a generic construction of extended IDTR as follows.

- Gen: On input 1^k, run $\mathcal{E}.\mathsf{Setup}(1^k)$ to generate a master key pair (mpk, msk) and set $\mathcal{D}_{\mathcal{R}} = mpk$ and $mtd_{\mathcal{R}} = msk$.

- Sample: On input $D_{\mathcal{R}}$ and id, randomly select $m \in \mathcal{MSP}$ and $s \in \{0,1\}^{\ell(k)}$, compute $r = f(s)$, and run $\mathcal{E}.\mathsf{Enc}(\mathcal{D}_{\mathcal{R}}, id, m; r)$ to generate a ciphertext e of m under the identity id. Store $(id, c, d) = (id, (e), (m, s))$ into a sampling list List and return (c, d).
- Extract: Given $mtd_{\mathcal{R}}$ and id, run $\mathcal{E}.\mathsf{Extract}(mtd_{\mathcal{R}}, id)$ to generate the corresponding private key sk_{id} with respect to the identity id, and return $td_{\mathcal{R}_{id}} = sk_{id}$.
- Invert: Given $td_{\mathcal{R}_{id}}$ and c, run $\mathcal{E}.\mathsf{Dec}(\mathcal{D}_{\mathcal{R}}, td_{\mathcal{R}_{id}}, c)$ to get the plaintext m, and return $\hat{d} = (td_{\mathcal{R}_{id}}, m)$.
- Reveal: Given $c \in \mathcal{L}_{\mathcal{R}_{id}}$, check if there is an entry for c in the sampling list List $= \{(id, c, d)\}$. If so, return the corresponding d; otherwise return $\perp$.
- CheckS: For any pair (c, d) output by algorithm Sample on input $\mathcal{D}_{\mathcal{R}}$ and id, we have that $(c, d) = ((e), (m, s))$. Check if $\mathcal{E}.\mathsf{Enc}(D_{\mathcal{R}}, id, m; f(s)) = e$. If so, return 1 (accept); otherwise return 0 (reject).
- CheckI: For any $(c, \hat{d}) \in \mathcal{R}_{id}$, where $\hat{d} \leftarrow \mathsf{Invert}_{td_{\mathcal{R}_{id}}}(c)$, we have that $(c, \hat{d}) = ((e), (sk_{id}, m))$. Check if $m = \mathcal{E}.\mathsf{Dec}(D_{\mathcal{R}}, sk_{id}, e)$. If so, return 1 (accept); otherwise, return 0 (reject).

Theorem 2. *The above scheme is a secure extended IDTR scheme, provided that the underlying IBE scheme $\mathcal{E}$ is IND-ID-CPA secure, and function f is one-way.*

Proof. For the sake of completeness of underlying ID-based Encryption schemes, we provide all the proofs of One-wayness, Soundness and Hiding here.

One-wayness: If the above scheme is not one-way, namely, there is a PPT algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which breaks the one-wayness property with non-negligible probability ϵ, we then construct a PPT algorithm $\mathcal{B}$ to break the IND-ID-CPA-security of the underlying encryption scheme $\mathcal{E}$ with non-negligible probability as well.

After obtaining system parameters and master public key mpk from its challenger, $\mathcal{B}$ sets $\mathcal{D}_{\mathcal{R}} = mpk$ and runs $\mathcal{A}$ on input $\mathcal{D}_{\mathcal{R}}$. To answer $\mathcal{A}_1$'s Extract query on id, $\mathcal{B}$ forwards this query to its own Extract oracle, and forwards the answer sk_{id} as $td_{\mathcal{R}_{id}}$ back to $\mathcal{A}$. After $\mathcal{A}_1$'s Extract query phase is over, $\mathcal{B}$ randomly selects an id^* which was not queried by $\mathcal{A}_1$, along with two random messages $m_0, m_1 \in \mathcal{MSP}$. It sends id^*, m_0, m_1 to its own challenger. After receiving a ciphertext e^* of either m_0 or m_1, $\mathcal{B}$ sets $c = (e^*)$, and feeds (id^*, c) to $\mathcal{A}_2$. Again, $\mathcal{B}$ needs answer $\mathcal{A}_2$'s Extract queries. It acts the same as in answering $\mathcal{A}_1$'s Extract queries with the only exception that if the query input id is equal to id^*, $\mathcal{B}$ aborts the simulation and outputs a random bit.

Finally, $\mathcal{A}_2$ outputs a $\hat{d} = (\hat{d}_1, \hat{d}_2)$. Then $\mathcal{B}$ computes $b_0 = \mathsf{CheckI}_{\mathcal{D}_{\mathcal{R}}, id^*}(c, \hat{d})$ and $b_1 = \mathsf{CheckS}_{\mathcal{D}_{\mathcal{R}}, id^*}(c, \hat{d})$. If $b_0 = 1$, $\mathcal{B}$ could easily get the corresponding private key sk_{id^*} of identity id^* as well as the plaintext m' of e^*, such that $m' = \mathcal{E}.\mathsf{Dec}(mpk, sk_{id^*}, e^*)$. It's guaranteed by the correctness of $\mathcal{E}$ that m' must be either m_0 or m_1. Thus, $\mathcal{B}$ can output the right bit b. Otherwise, if $b_1 = 1$, we have that $e^* = \mathcal{E}.\mathsf{Enc}(mpk, id^*, \hat{d}_1; f(\hat{d}_2))$. Again, guaranteed by the correctness of $\mathcal{E}$, $\hat{d}_1$ is either m_0 or m_1. Thus $\mathcal{B}$ can know a bit b such that

$m_b = \hat{d}_1$. It outputs b and wins in the IND-ID-CPA game. If both b_0 and b_1 are 0, $\mathcal{B}$ simply flips a coin, and outputs the outcome. Obliviously, if $\mathcal{A}$ succeeds, $\mathcal{B}$ also succeeds. Therefore, the probability that $\mathcal{B}$ wins in the IND-ID-CPA game is at least $\epsilon + \frac{1}{2}(1-\epsilon) = \frac{1}{2} + \frac{1}{2}\epsilon$, which is non-negligibly greater than one-half.

Soundness: The soundness is guaranteed by the correctness of the underlying encryption scheme $\mathcal{E}$. That is, for any valid ciphertext e with respect to any identity id, the owner of the corresponding private key sk_{id} can always decrypt e to the original message m.

Hiding: If the above scheme is not hiding, that is, there is a PPT algorithm $\mathcal{A}$ which can break the hiding property with non-negligible probability ϵ, then we can construct another PPT algorithm $\mathcal{B}$ to break the one-wayness of function f with non-negligible probability as well.

On input $y = f(x)$ for some string $x \in \{0,1\}^{\ell(k)}$, $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine. Suppose that $\mathcal{A}$ issues at most q_S queries to $\mathcal{O}_{\mathsf{Sample}}$ and at most q_R queries to $\mathcal{O}_{\mathsf{Reveal}}$. Implicitly, we have that $q_S > q_R$. $\mathcal{B}$ randomly chooses $i \in \{1, 2, \cdots, q_S\}$, and then simulates oracles $\mathcal{O}_{\mathsf{Extract}}$ and $\mathcal{O}_{\mathsf{Reveal}}$ for $\mathcal{A}$ as follows:

$\mathcal{O}_{\mathsf{Sample}}$**:** Suppose that this is the j-th query. On input id_j, $\mathcal{B}$ randomly selects $m \in \mathcal{MSP}$. If $j = i$, $\mathcal{B}$ sets $r = y$ and $d = (m, \perp)$; otherwise, $\mathcal{B}$ randomly selects $s \in \{0,1\}^{\ell(k)}$, computes $r = f(s)$ and set $d = (m, s)$. It computes $e \leftarrow \mathcal{E}.\mathsf{Enc}(\mathcal{D}_{\mathcal{R}}, id_j, m; r)$ and sets $c = (e)$. $\mathcal{B}$ stores (id_j, c, d) into a sampling list List, and returns c. Note that in this way $\mathcal{B}$ perfectly simulates $\mathcal{O}_{\mathsf{Sample}}$'s answers.

$\mathcal{O}_{\mathsf{Reveal}}$**:** On input id and $c = (e)$, $\mathcal{B}$ searches its sampling list List for an entry for (id, c). If there is no List or no such an entry $(id, c, d = (d_1, d_2))$, $\mathcal{B}$ returns $\perp$; otherwise, if $d_2 = \perp$, $\mathcal{B}$ aborts; otherwise, it returns d.

Finally, $\mathcal{A}$ outputs $(id^*, (c^*, d^*))$ where $d^* = (d_1^*, d_2^*)$. If $\mathcal{A}$ wins in the Hiding game, we have $c^* \in Query(\mathcal{A}, \mathcal{O}_{\mathsf{Sample}})$, $c^* \notin Query(\mathcal{A}, \mathcal{O}_{\mathsf{Reveal}})$ and $\mathsf{CheckS}_{\mathcal{D}_{\mathcal{R}}, id^*}(c^*, d^*) = 1$. It implies that $c^* = \mathcal{E}.\mathsf{Enc}(\mathcal{D}_{\mathcal{R}}, id^*, d_1^*; f(d_2^*))$. If $\mathcal{B}$'s guess is correct, namely, c^* is returned in the answer to the i-th Sample query, then by the $\mathsf{injective}$ property of $\mathcal{E}$, we have that $f(d_2^*) = y$. Thus, d_2^* is a pre-image of y. The probability that $\mathcal{B}$ succeeds in guessing i is at least $\frac{1}{q_S}$. If $\mathcal{A}$ breaks the Hiding property with probability ϵ, then $\mathcal{B}$ breaks the one-wayness of f with probability at least $\frac{\epsilon}{q_S}$, which is non-negligible. This is a contradiction to the one-wayness of f.

6.3 Extended Time Capsule Signature

The Ver function in time capsule signature can also be separated into two functions accordingly: VerP is to verify the prehatched signature, VerH is to verify the hatched signature. The generic construction of time capsule signature based on IDTR can then be modified as follows:

- VerP: For a given prehatched signature $\sigma_t = (s, c, d)$ on m, a verifier checks if $\mathsf{CheckS}_{tpk,t}(c, d)$ outputs 1 and $\mathsf{Verify}_{upk}(m \| c \| t, s)$ outputs 1. If both of the verifications are correct, output 1; otherwise, output 0.

– VerH: For a given hatched signature $\sigma_t = (s, c, \hat{d})$ on m, the verifier compares the current time with t. If the current time is smaller than t, it returns $\perp$ indicating that hatching cannot be done at the moment. Otherwise, the verifier determines if $\mathsf{CheckI}_{tpk,t}(c, \hat{d})$ outputs 1 and $\mathsf{Verify}_{upk}(m\|c\|t, s)$ outputs 1. If both of the verifications are correct, output 1; otherwise, output 0.

In the construction of [13], the Time Server should be fully trusted and it is assumed that the Time Server would not collude with any malicious user and release some time trapdoor z_t before t. Otherwise, there is no way to distinguish whether a signature is pre-hatched by the actual signer or hatched by a malicious Time Server. In our distinguishable time capsule signature, we make this act of a malicious Time Server distinguishable. Below is the formal security model. Let $k \in \mathbb{N}$ be a security parameter.

Game IV: Let $\mathcal{S}_{IV}$ be the game simulator.

1. $\mathcal{S}_{IV}$ executes $\mathsf{TSSetup}(1^k)$ to get (tpk, tsk) and $\mathsf{UserSetup}(1^k)$ to get (upk, usk).
2. $\mathcal{S}_{IV}$ runs $\mathcal{A}_{IV}$ on upk, tpk and tsk. During the simulation, $\mathcal{A}_{IV}$ can make queries onto TSig, and PreHatch.
3. $\mathcal{A}_{IV}$ is to output (m^*, t^*, σ^*).

$\mathcal{A}_{IV}$ *wins* if $\mathsf{VerP}(m^*, \sigma^*, upk, tpk, t^*) = 1$, and $\mathcal{A}_{IV}$ has never queried $\mathsf{PreHatch}(m^*, t^*, \cdot)$.

A time capsule signature scheme is secure in **Game IV** if for all PPT algorithm $\mathcal{A}_{IV}$, it is negligible for $\mathcal{A}_{IV}$ to win the game.

Now we prove the security of our proposed time capsule signature scheme against Game IV.

Theorem 3. *The extended time capsule signature scheme is secure in Game IV if the underlying extended IDTR scheme has the Hiding property, and the standard signature scheme is existentially unforgeable against adaptive chosen message attacks (euf-cma) [16].*

Proof. To show security against Game IV, we construct an adversary $\mathcal{B}$ which can compromise Hiding of the extended IDTR with non-negligible advantage if $\mathcal{A}_{IV}$ can non-negligibly forge a prehatched signature σ^*. Let (m^*, t^*, σ^*) be a successful forgery by $\mathcal{A}_{IV}$, where $\sigma^* = (s^*, c^*, d^*)$. Note that it has overwhelming probability that $\mathcal{A}_{IV}$ obtained the corresponding time capsule signature σ'^* from oracle TSig. This is because of the euf-cma assumption of the underlying standard signature scheme.

The game between the challenger of the extended IDTR Hiding game and adversary $\mathcal{B}$ starts when the challenger generates $D_{\mathcal{R}}$ and $mtd_{\mathcal{R}}$ by running $\mathsf{Gen}(1^k)$, and then gives $D_{\mathcal{R}}$ and $mtd_{\mathcal{R}}$ to $\mathcal{B}$. $\mathcal{B}$ then interacts with $\mathcal{A}_{IV}$ as follows:

$\mathcal{B}$ gets a random public/private key pair $(pk, sk) \leftarrow \mathsf{Set}(1^k)$, sets $(upk, usk) = (pk, sk)$, $(tpk, tsk) = (D_{\mathcal{R}}, mtd_{\mathcal{R}})$, and gives (tpk, tsk, upk) to $\mathcal{A}_{IV}$.

$\mathcal{B}$ manages a list $L = \{(m_i, t_i, s_i, c_i, d_i)\}$ for answering $\mathcal{A}_{IV}$'s queries to PreHatch. Let q_{TSig} and q_{PreH} be the total number of TSig and PreHatch queries

made by $\mathcal{A}_{IV}$, respectively, and r be the random number chosen by $\mathcal{B}$ in the interval of $[1, q_{\mathsf{TSig}}]$. $\mathcal{B}$ responds to the i-th TSig query (m_i, t_i) as follows:

- If $i = r$, $\mathcal{B}$ queries to its challenger on the Sample oracle on t_r and receives a random commitment $c \in R_{t_r}$. $\mathcal{B}$ sets $c_r = c$ and computes $s_r = \mathsf{Sig}_{sk}(m_r \| c_r \| t_r)$. $\mathcal{B}$ returns $\sigma'_{t_r} = (s_r, c_r)$ to $\mathcal{A}_{IV}$ and stores $(m_r, t_r, s_r, c_r, \perp)$ in L.
- If $i \neq r$, $\mathcal{B}$ gets a random commitment/witness pair $(c_i, d_i) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}$ and $s_i \leftarrow \mathsf{Sig}_{sk}(m_i \| c_i \| t_i)$. $\mathcal{B}$ returns $\sigma'_{t_i} = (s_i, c_i)$ to $\mathcal{A}_{IV}$ and stores $(m_i, t_i, s_i, c_i, d_i)$ in L.

To simulate $\mathsf{PreHatch}$ on query (m_i, t_i, s_i, c_i), $\mathcal{B}$ checks if the query is in the list L. If (m_i, t_i, s_i, c_i) is L, and equal to (m_r, t_r, s_r, c_r), $\mathcal{B}$ aborts. If (m_i, t_i, s_i, c_i) is in L, and not equal to (m_r, t_r, s_r, c_r), $\mathcal{B}$ obtains d_i from L and gives a prehatched signature $\sigma_{t_i} = (s_i, c_i, d_i)$ to $\mathcal{A}_{IV}$. If (m_i, t_i, s_i, c_i) is not in L, since $\mathcal{A}_{II}$ does not know usk and this case implies that s_i is not generated by $\mathcal{B}$ on $m_i \| c_i \| t_i$, due to the euf-cma assumption of the underlying standard signature scheme, it is negligible to have s_i be valid. Hence this case will happen with negligible chance. For this case, $\mathcal{B}$ returns $\perp$.

When $\mathcal{A}_{IV}$ outputs the forgery (m^*, t^*, σ^*) where $\sigma^* = (s^*, c^*, d^*)$, $\mathcal{B}$ determines if the forgery passes CheckS, and $(m^*, t^*, s^*, c^*) = (m_r, t_r, s_r, c_r)$. If so, $\mathcal{B}$ outputs d^*. Otherwise, it chooses a value d randomly and outputs d. The probability that $\mathcal{B}$ does not abort during the simulation and has the right guess of r is at least $1/q_{\mathsf{TSig}}$ since r is randomly chosen (*). Therefore, if $\mathcal{A}_{IV}$ forges with success probability at least ϵ, $\mathcal{B}$ succeeds in breaking the Hiding property of the extended IDTR with probability at least $\epsilon/q_{\mathsf{TSig}}$.

(*) Without loss of generality, we assume that each TSig query is distinct and each $\mathsf{PreHatch}$ is also distinct, and $q_{\mathsf{PreH}} \leq q_{\mathsf{TSig}}$. The probability that $\mathcal{A}_{IV}$ outputs a forgery (m^*, t^*, s^*, c^*) which passes CheckS but not in the list L is negligible due to the euf-cma assumption of the underlying standard signature scheme. $\mathcal{B}$ does not abort when answering the first $\mathsf{PreHatch}$ query is at least $(1 - 1/q_{\mathsf{TSig}})$. It does not abort when answering the second $\mathsf{PreHatch}$ query is at least $(1 - 1/q_{\mathsf{TSig}}) \times (1 - 1/(q_{\mathsf{TSig}} - 1))$. Finally we get

$$
\begin{aligned}
& Pr[\mathcal{B} \text{ does not abort}] \\
\geq\ & (1 - \frac{1}{q_{\mathsf{TSig}}}) \times (1 - \frac{1}{(q_{\mathsf{TSig}} - 1)}) \times \cdots \times (1 - \frac{1}{q_{\mathsf{TSig}} - q_{\mathsf{PreH}} + 1}) \\
=\ & \frac{q_{\mathsf{TSig}} - 1}{q_{\mathsf{TSig}}} \times \frac{q_{\mathsf{TSig}} - 2}{q_{\mathsf{TSig}} - 1} \times \cdots \times \frac{q_{\mathsf{TSig}} - q_{\mathsf{PreH}}}{q_{\mathsf{TSig}} - q_{\mathsf{PreH}} + 1} \\
=\ & \frac{q_{\mathsf{TSig}} - q_{\mathsf{PreH}}}{q_{\mathsf{TSig}}}
\end{aligned}
$$

And $\mathcal{B}$ makes the right guess of r in the remaining $q_{\mathsf{TSig}} - q_{\mathsf{PreH}}$ tuples is $1/(q_{\mathsf{TSig}} - q_{\mathsf{PreH}})$. Thus, the probability that $\mathcal{B}$ does not abort during the simulation and makes the right guess of r is at least $1/q_{\mathsf{TSig}}$.

7 Conclusion

Time Capsule Signature is a promising technique for various E-Commerce applications. In this paper, we improve the security model of time capsule signature, construct a generic and provably secure time capsule signature scheme based on a new primitive called identity-based trapdoor relation (IDTR), and show that IDTR can be implemented efficiently by proposing two instantiations. We believe that the IDTR itself is of independent interest and may be implemented by other techniques. We leave these as our further investigations.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.
2. M. Bellare and S. Goldwasser. Encapsulated key escrow. Technical Report 688, MIT/LCS/TR, 1996.
3. M. Bellare and S. Goldwasser. Verifiable partial key escrow. In *ACM Conference on Computer and Communications Security*, pages 78–91, 1997.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
5. I. F. Blake and A. C.-F. Chan. Scalable, server-passive, user-anonymous timed release public key encryption from bilinear pairing. In *ICDCS*, 2005.
6. D. Boneh and X. Boyen. Efficient selective-id secure identity based encryption without random oracles. In *Proc. EUROCRYPT 2004*. Springer-Verlag, 2004. LNCS.
7. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Proc. CRYPTO 2001*, pages 213–229. Springer-Verlag, 2001. LNCS 2139.
8. D. Boneh, C. Gentry, H. Shacham, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt'03*, pages 416–432. Spinger, 2003. LNCS.
9. D. Boneh and M. Naor. Timed commitments. In *Proc. CRYPTO 2000*, page 236. Springer-Verlag, 2000. LNCS 1880.
10. L. Chen, K. Harrison, N. Smart, and D. Soldera. Applications of multiple trust authorities in pairing based cryptosystems. In *Infrastructure Security Conference 2002*, pages 260–275. Spinger-Verlag, 2002. LNCS 2437.
11. J. H. Cheon, N. Hopper, Y. Kim, and I. Osipkov. Timed-release and key-insulated public key encryption. Cryptology ePrint Archive, Report 2004/231, 2004.
12. Y. Dodis and L. Reyzin. Breaking and repairing optimistic fair exchange from PODC 2003. In *ACM Workshop on Digital Rights Management (DRM)*, Oct. 2003.
13. Y. Dodis and D. Yum. Time capsule signature. In *Financial Cryptography and Data Security 2005*, pages 57–71. Springer-Verlag, 2005. LNCS 3570.
14. J. A. Garay and M. Jakobsson. Timed release of standard digital signatures. In *Financial Cryptography and Data Security 2002*, pages 168–182. Spinger-Verlag, 2002. LNCS 2357.
15. J. A. Garay and C. Pomerance. Timed fair exchange of standard signatures. In *Financial Cryptography and Data Security 2003*, pages 190–207. Springer-Verlag, 2003. LNCS 2742.

16. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, Apr. 1988.
17. T. C. May. Timed-release crypto, 1993. www.cyphernet.org/cyphernomicon/chapter14/14.5.html.
18. M. C. Mont, K. Harrison, and M. Sadler. The HP time vault service: Exploiting IBE for timed release of confidential information. In *WWW*, 2003.
19. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Asiacrypt01*, pages 552–565. Spinger-Verlag, 2001. LNCS /2248.
20. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report 684, MIT/LCS/TR, 1996.
21. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. CRYPTO 84*, pages 47–53. Springer, 1984. LNCS 196.
22. B. Waters. Efficient identity-based encryption without random oracles. In *Proc. EUROCRYPT 2005*, pages 114–127. Springer-Verlag, 2005. LNCS 3494.
23. M. Zhang, G. Chen, J. Li, L. Wang, and H. Qian. A new construction of time capsule signature. Cryptology ePrint Archive, Report 2006/113, 2006. http://eprint.iacr.org.