Time Capsule Signature

Yevgeniy Dodis[1] and Dae Hyun Yum[1,2]

[1]Dept. of Computer Science, New York University, NY, USA
[2]Dept. of Electronic and Electrical Eng., POSTECH, Pohang, Korea
{dodis, dhyum}@cs.nyu.edu

Abstract. We introduce a new cryptographic problem called *time capsule signature*. Time capsule signature is a 'future signature' that becomes valid from a specific future time t, when a trusted third party (called *Time Server*) publishes some trapdoor information associated with the time t. In addition, time capsule signature should satisfy the following properties:

(1) If the signer wants, she can make her time capsule signature effective before the pre-defined time t.
(2) The recipient of 'future signature' can verify right away that the signature will become valid no later than at time t.
(3) Time Server need not contact any user at any time, and in fact does not need to know anything about the PKI employed by the users.
(4) Signatures completed by the signer before time t are indistinguishable from the ones completed using the Time Server at time t.

We provide the rigorous definition of time capsule signature and the generic construction based on another new primitive of independent interest, which we call *identity-based trapdoor hard-to-invert relation* (ID-THIR). We also show an efficient construction of ID-THIRs (and, hence, time capsule signatures) in the random oracle model, and a less efficient construction in the standard model.

If the time t is replaced by a specific event, the concept of time capsule signature can be generalized to *event capsule signature*.

1 Introduction

1.1 Time Capsule Signature

In an ordinary signature scheme, the validity of a signature value is determined at the point of signature generation and never changes (unless the signer's public key is revoked). Users cannot generate the so-called 'future signature' which is not currently valid but becomes valid from a future time t. A naive way to achieve this is signing with a statement such as 'the signature of message m becomes valid from time t.' This, however, has several drawbacks. First, and least serious, the verifier is required to be aware of the current time. When time is generalized to arbitrary events (i.e., 'the signature of m becomes valid if the event e happens'), this becomes even more problematic. More seriously, however, in the naive solution the signer herself loses control over the validity of the future signature, i.e., even the real signer cannot make her signature valid before time t. This means that either the signer has to wait until time t — which could be undesirable in certain situations (e.g., if the borrower wants to quickly repay her debt before the actual due date to improve her credit history) — or the signer can issue a new, independent signature of m before time t. The latter solution, however, can also be undesirable in certain situations. First, in case the message m carries some monetary value, the signer needs to make sure that no "double spending" occurs (i.e., to somehow revoke the original signature, so that it does not become valid at time t). Second, the verifier now knows whether the message m was signed in the 'future' or 'regular' way, which seems to be unnecessary in most situations.

Therefore, we would like a solution where the signer can issue a future signature so that at least the following properties are satisfied:

(1) At the time of creation, the recipient is sure that the signature will become valid by time t, even if the signer refuses to cooperate after she produces the future signature.
(2) The legal signer can make the future signature valid at any time after the initial creation.
(3) Irrespective of whether the signer validated the signature earlier, or it became "automatically valid" at time t, the resulting signatures are indistinguishable. In other words, the verifier after time t cannot tell the lower-level details of how the signature became valid.

Of course, it is also crucial to specify the mechanism under which the signature can be "automatically" completed at time t (which we call "hatching" as opposed to "pre-hatching" which can be done by the signer at any time). As we remarked, we cannot just make it valid at time t, since this requires the verifier to "know" the current time, and, more importantly, will not make the hatching indistinguishable from pre-hatching. Another option would be to use some "time-release" primitive, such as timed signature [10], where the verifier knows that by investing some intensive computation, he can complete a future signature within some pre-specified time, even if the signer refuses to cooperate. However, this option is only approximate (i.e., the verifier can open the signature roughly by time t depending on its computational capabilities), and, more importantly, forces the verifier to invest a considerable computational effort the moment the future signature was generated.

Finally, we can follow the approach of optimistic fair exchange protocols [1, 2, 16], where an "off-line" arbitrator (a trusted third party) can complete the signer's partial signature into the full signature, shall the signer refuse to cooperate. In particular, so called verifiably committed signatures of [16] seem to be ideally suited for this task. The main drawback of this solution is that the arbitrator, although only involved if the signer refuses to cooperate (say, before time t), has to be involved in a message-by-message manner. Thus, in our scenario of future signatures, — where by default the signer will not pre-hatch her signature, — the arbitrator would have to literally complete almost every signature separately. The latter, of course, makes the arbitrator quite "on-line" and whole paradigm very unattractive for our application.

Instead, we introduce *time capsule signatures*, where the arbitrator (which we call the *Time Server*)

(1) Does not ever need to contact users, know about the particular format of their signature, or be involved in any signature resolution protocols.
(2) At the beginning of time period t, outputs a single message Z_t, which automatically allows anybody to complete any future signature set to hatch at time t.

More specifically, time capsule signature is a 'future signature' that becomes valid from a specific future time t, when a trusted third party (called *Time Server*) publishes some trapdoor information associated with the time t. When Alice gives Bob her time capsule signature σ'_t for a future time t, Bob can verify that Alice's time capsule signature will become valid from the time t. In addition, if Alice wishes, she can make her time capsule signature effective before the predefined time t. The assumption on Time Server is minimal, in that Time Server only publishes some information at the beginning of each time period and need not contact any user at any time. Finally, the concept of time capsule signature can be generalized to *event capsule signature*, where *Event Server* issues the notification information of specific events. The event capsule signature becomes valid if a specific event happens or the signer makes valid before the event occurs.

1.2 Our Contribution

We provide the rigorous definition of time (or event) capsule signature and the generic construction based on another new primitive of independent interest, which we call *identity-based trapdoor hard-to-invert relation* (ID-THIR). Intuitively, ID-THIR is given by a family $\mathcal{R}$ of relations R_{id}, where (1) it is easy to sample a random pair $(c, d) \in R_{id}$ and verify if the pair (c, d) belongs to R_{id}; (2) for each identity id, there exists a trapdoor td_{id}, which allows one to compute a random d corresponding (w.r.t. R_{id}) to any given c (The trapdoor td_{id}'s can be efficiently computed from a single "master key" $\mathsf{mtd}_{\mathcal{R}}$); (3) without the trapdoor td_{id}, it is hard to find a matching d corresponding (w.r.t. R_{id}) to a randomly sampled c, even if one knows many trapdoors corresponding to identities $id' \neq id$.

Our construction of time (or event) capsule signatures from ID-THIR is very natural: the future signature of m is $(\mathbf{Sig}(m||c||t), c)$, while the full hatched signature is $(\mathbf{Sig}(m||c||t), c, d)$, where '$\mathbf{Sig}$' is any ordinary signature, 'm' a message, '$\|$' the concatenation and (c, d) a random lock/proof pair corresponding to the "identity" equal to time t (or event e). The legal signer would sample (c, d) and remember d for pre-hatching, while the Time (or Event) Server would periodically publish the trapdoors td_t (or td_e) which would allow anyone to hatch the signature by computing the corresponding d from c. Moreover, hatching and pre-hatching would look the same, by the security properties of the ID-THIR scheme.

Finally, we give a generic construction of ID-THIR (and, therefore, time/event capsule signature). In the standard model, the construction is mainly of theoretical interest, as it relied on non-interactive witness indistinguishable proofs of knowledge. Nevertheless, it shows that our primitives exist if trapdoor one-way permutations exist. On a practical front, we give several very efficient implementations of ID-THIR in the random oracle model. Generically, we show that in the random oracle model one can construct our primitives from mere one-way functions. Concretely, we show very efficient instantiations based on RSA or discrete log in the Gap Diffie-Hellman groups [26, 8].

1.3 Related Work

As we pointed out, there are two main lines of work related to time capsule signature, depending on whether or not the trusted third party is involved.

The first approach, which is that of timed-release cryptography, is to ensure that the encryption, commitment or signature can be opened in a brute force way by solving some time-consuming, but computationally feasible problem. For example, Dwork and Naor [17] used such moderately hard functions in order to deter abuse of resources, such as spamming. Bellare and Goldwasser [3, 4] suggested "(verifiable) time capsules"[1] for key escrowing in order to deter widespread wiretapping. There, the main issue is the verification at escrow time that the right key will be recovered. Rivest, Shamir and Wagner [29] suggested "time-lock puzzle," where the goal is to design "inherently sequential puzzles" which are resistant to parallel attacks. However, they did not address verifiability at escrow time. The latter was formally addressed by Boneh and Naor [10], who defined (verifiable) timed commitments. As one of their applications, they get an analog of our time capsule signature (termed "timed signature"), where the future signature can either be opened by the signer, or by the recipient — the latter if the recipient solves a moderately hard problem. More recent advances were made by [21, 22].

The second approach, based on the trusted third party, has two main flavors: optimistic fair exchange of digital signatures, and identity-based future encryption. In the former case, the server needs to resolve all individual signatures where the signer refused to validate the signature (say, by a given time t).[2] Representative examples include [1, 2, 11, 16]. In contrast, in our model we insist that users do not communicate ever with the trusted server.

In the case of future encryption [7, 6, 27], the main problem addressed was that the sender wants to ensure that the message would remain hidden before the Time Server would publish the corresponding trapdoor. However, this is orthogonal to our model, where we want to "encrypt" a *signature* on a *public* message. Thus, we do not need to hide the message (and can even leak partial

[1] Which should not be confused with our time capsule signatures, which are totally different.

[2] In fact, in some of the solutions the clients additionally need to either register their keys with the server, or have an interactive resolution protocol.

information about the full signature, as long as the full signature is hidden). On the other hand, we have to resolve two crucial complications not present in the above scenario: (1) the future signature has to be verifiable right away, to ensure the recipient it will be successfully completed at time t; (2) the sender can pre-hatch the signature in a manner indistinguishable from the regular hatching at time t. Not surprisingly, the solutions above all utilized some kind of identity-based encryption, and do not seem to be useful for our time-capsule signatures.

2 Primitives

2.1 Σ-protocol

A Σ-protocol [15] is an efficient 3-round two-party protocol between the prover and the verifier on a common input $x \in \mathcal{L}_R$, where $\mathcal{L}_R$ is a language for an NP relation R. Besides x, a valid NP-witness w for x is also given to the prover as a private input. The prover first sends a commitment message a to the receiver. After receiving the commitment message a, the verifier sends a challenge message b to the prover. Finally, the prover sends a response message z to the verifier who decides to output 1 (accept) or 0 (reject) based on the input x and the transcript $\pi = \{a, b, z\}$. The transcript π is valid if the verifier outputs 1 (accept). A binary Σ-protocol is a special case of Σ-protocol where the challenge message takes only a binary value (0 or 1).

A Σ-protocol should satisfy three properties: correctness, special soundness, and special (honest-verifier) zero-knowledge. Correctness property states that for all $x \in \mathcal{L}_R$ and all valid witnesses w for x, if the prover and the verifier follow the protocol honestly, the verifier must output 1 (accept). Special soundness property says that there is an efficient extraction algorithm (called a knowledge extractor) Ext that on input $x \in \mathcal{L}_R$ and two valid transcripts π_1, π_2 with the same commitment message outputs z such that $(x, z) \in R$. Special zero-knowledge property says that there is an efficient simulation algorithm (called a simulator) Sim that on input $x \in \mathcal{L}_R$ and any challenge message b, outputs a valid transcript $\pi' = \{a', b, z'\}$. Moreover, the distribution of (a', z') is computationally indistinguishable from the corresponding distribution on (a, z) produced by the prover knowing a valid witness w for x and the verifier. This is true even if the distinguisher knows the witness w.

A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function, if there exists a polynomial time algorithm which computes $f(x)$ correctly for all x and the following probability is negligible for all PPT (Probabilistic Polynomial Time) algorithm A: $\Pr(f(x') = y \mid x \leftarrow \{0,1\}^k; y = f(x); x' \leftarrow A(y, 1^k))$. It is known that any language in NP has a Σ-protocol if one-way functions exist [23, 18]. Of course, specific languages can have much more efficient Σ-protocols. A Σ-protocol can also be transformed into a signature scheme by using the Fiat-Shamir heuristic [20]. To sign a message m, the legal signer produces a valid transcript $\pi = \{a, b, z\}$ of the Σ-protocol, where $b = H(a, m)$ and $H(\cdot)$ is a cryptographic hash function modelled as a random function. The signature scheme obtained by applying the Fiat-Shamir heuristic to the Σ-protocol is secure in the random oracle model [5, 31]. It is also known that the Fiat-Shamir heuristic gives a non-interactive proof of knowledge in the random oracle model (i.e., the witness can be extracted by rewinding the adversary).

If there are two Σ-protocols, i.e., Σ_1 for R_1 and Σ_2 for R_2, we can construct another Σ-protocol Σ_{OR} (called OR-proof) [15] which allows the prover to show that given two inputs x_1, x_2, he knows w such that either $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$ without revealing which is the case (called the witness indistinguishability property [19]). By applying the Fiat-Shamir heuristic to the OR-proof Σ_{OR}, we get a signature scheme (called OR-signature) secure in the random oracle model such that a valid signature can be generated by the signer who knows a valid witness w corresponding to either of the two inputs x_1, x_2. It is known that the Fiat-Shamir heuristic does not affect the witness indistinguishability property of the Σ-protocol.

2.2 Identity-Based Trapdoor Hard-to-Invert Relation

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., $\mathcal{L}_R = \{\alpha \mid \exists \beta\ [(\alpha, \beta) \in R]\}$. 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.

We also assume that the membership in $f(X)$ can be efficiently determined for a (trapdoor) one-way function $f : X \to Y$.

Definition 1. *An* identity-based trapdoor hard-to-invert relation (ID-THIR) *is a set of relations $\mathcal{R} = \{R_{id} \mid id \in I_{\mathcal{R}}\}$, where each relation R_{id} is trapdoor hard-to-invert relation (i.e., sampling a random lock/proof pair $(c, d) \in R_{id}$ is easy but finding a proof for a given lock is difficult without knowing the trapdoor* td_{id}*) and there is a master trapdoor* $\mathsf{mtd}_{\mathcal{R}}$ *for extracting the trapdoor* td_{id} *of each relation R_{id}.* ID-THIR *can also be specified by 5-tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Sample}, \mathsf{Check}, \mathsf{Extract}, \mathsf{Invert})$ *such that:*

- Gen*. This algorithm is used to generate $\mathcal{R} = \{R_{id} \mid id \in I_{\mathcal{R}}\}$, where $I_{\mathcal{R}}$ is a finite set of indices.* $\mathsf{Gen}(1^k)$ *returns $D_{\mathcal{R}}$ (the description of $\mathcal{R}$) and* $\mathsf{mtd}_{\mathcal{R}}$ *(the master trapdoor).*
- Sample*. This sampling algorithm takes $(D_{\mathcal{R}}, id)$ as input and* $\mathsf{Sample}_{D_{\mathcal{R}}}(id)$ *returns a random lock/proof pair $(c, d) \in R_{id}$.*
- Check*. This algorithm is used to check the validity of the proof. If $(c, d) \in R_{id}$, then* $\mathsf{Check}_{D_{\mathcal{R}},id}(c, d)$ *returns 1 (accept). Otherwise, it returns 0 (reject).*
- $\mathsf{Extract}$*. This algorithm is used to extract the trapdoor of each relation by using* $\mathsf{mtd}_{\mathcal{R}}$*.* $\mathsf{Extract}_{\mathsf{mtd}_{\mathcal{R}}}(id)$ *returns the trapdoor* $\mathsf{td}_{R_{id}}$ *of the relation R_{id}.*
- Invert*. This algorithm is used to find a proof 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 proof d such that $(c, d) \in R_{id}$.*

Let $(c, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(id)$ *and* $\widetilde{d} \leftarrow \mathsf{Invert}_{\mathsf{td}_{R_{id}}}(c)$*. Correctness property states that* $\mathsf{Check}_{D_{\mathcal{R}},id}(c, d) = 1$ *and* $\mathsf{Check}_{D_{\mathcal{R}},id}(c, \widetilde{d}) = 1$*, and ambiguity property* *states that (c, d) and $(c, \widetilde{d})$ are computationally indistinguishable, even if the distinguisher knows the master key $\mathsf{mtd}_{\mathcal{R}}$. Let O_{Extract} be the oracle simulating the trapdoor extraction procedure* Extract *and $Query(A, O_{\mathsf{Extract}})$ the set of queries an algorithm A asked to O_{Extract}. One-wayness property states that the following probability is negligible for all PPT algorithm $A = (A_1, A_2)$:*

$$\Pr[\,\mathsf{Check}_{D_{\mathcal{R}},id}(c, \widehat{d}) = 1 \;\wedge\; id \notin Query(A, O_{\mathsf{Extract}}) \mid (D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}}) \leftarrow \mathsf{Gen}(1^k);\; (id, h) \leftarrow A_1^{O_{\mathsf{Extract}}}(D_{\mathcal{R}});\; (c, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(id);\; \widehat{d} \leftarrow A_2^{O_{\mathsf{Extract}}}(D_{\mathcal{R}}, c, h)]$$

Soundness property states that the following probability is negligible for all algorithm B:

$$\Pr[\,R_{id} \in \mathcal{R} \;\wedge\; c \in \mathcal{L}_{R_{id}} \;\wedge\; \mathsf{Check}_{D_{\mathcal{R}},id}(c, \widetilde{d}) = 0 \mid (D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}}) \leftarrow \mathsf{Gen}(1^k);\; (c, id) \leftarrow B(D_{\mathcal{R}});\; \mathsf{td}_{R_{id}} \leftarrow \mathsf{Extract}_{\mathsf{mtd}_{\mathcal{R}}}(id);\; \widetilde{d} \leftarrow \mathsf{Invert}_{\mathsf{td}_{R_{id}}}(c)\,]$$

If ID-THIR *satisfies these four properties, we say that* ID-THIR *is secure.*

Construction. Each trapdoor hard-to-invert relation R_{id} in ID-THIR $\mathcal{R} = \{R_{id} \mid id \in I_{\mathcal{R}}\}$ looks like a trapdoor one-way function. However, there is an important difference: we can sample a random lock/proof pair $(c, d) \in R_{id}$ but may not necessarily be able to compute a lock c for a given proof d. Therefore, we can show that a trapdoor one-way function implies a trapdoor hard-to-invert relation but cannot prove the reverse direction. While the concept of ID-THIR also seems very general, the construction is not trivial. For example, it is not obvious whether identity-based encryption (IBE) [35, 7] implies ID-THIR or not, since IBE does not automatically guarantee the ambiguity property of ID-THIR.[3] Now, we provide our general construction of ID-THIR.

Theorem 1. *If there is a one-way function, there exists a secure* ID-THIR *in the random oracle model.*

Proof: Assume that there is a one way function $f : X \rightarrow Y$. We can build a secure signature scheme $(\mathbf{Set}, \mathbf{Sig}, \mathbf{Ver})$ from the one-way function f, since secure signatures exist if and only if one-way functions exist [32]. Let $\Sigma_{\mathbf{Sig}}$ be the Σ-protocol for the knowledge of a signature value $\mathbf{Sig}(m)$ on a common input $m \in M$ and Σ_f for the knowledge of a pre-image of a common input $f(x) \in f(X)$. If we denote by Σ_{OR} the OR-proof for $\Sigma_{\mathbf{Sig}}$ or Σ_f, we can obtain an OR-signature scheme $(\mathbf{Set}^{OR}, \mathbf{Sig}^{OR}, \mathbf{Ver}^{OR})$ by applying the Fiat-Shamir heuristic to Σ_{OR}. The OR-signature is secure in the random oracle model and $\mathbf{Set}^{OR}$ can be implicitly defined by $\Sigma_{\mathbf{Sig}}$ and Σ_f.

Now, we define the identity-based trapdoor hard-to-invert relation $\mathcal{R}_{OR} = \{R_m \mid m \in M\}$ where $R_m = \{(y, \pi) \mid y = f(x) \text{ for } x \in X,\ \pi \text{ is an OR-signature on } m\|f(x) \text{ for the knowledge of a pre-image of } f(x) \text{ or } \mathbf{Sig}(m)\}$ and the algorithms

[3] The decryption algorithm of IBE does not necessarily recover the temporary random number used in the encryption algorithm. For example, see [7].

(Gen, Sample, Check, Extract, Invert) as follows; Gen chooses $(pk, sk) \leftarrow \mathbf{Set}(1^k)$ and outputs $D_{\mathcal{R}_{OR}} = pk$, $\mathsf{mtd}_{\mathcal{R}_{OR}} = sk$ (technically, $D_{\mathcal{R}_{OR}}$ should also contain the one-way function f). We assume that message space M is known implicitly. On input $id = m$, Sample randomly chooses $x \in X$ and generates an OR-signature π for the knowledge of a pre-image of $f(x)$ or $\mathbf{Sig}(m)$. Sample outputs a lock/proof pair $(c, d) = (f(x), \pi)$. For a given $(id, c, d) = (m, f(x), \pi)$, Check verifies whether π is a valid OR-signature for a pre-image of $f(x)$ or $\mathbf{Sig}(m)$. Extract takes as input $id = m$ and outputs $\mathsf{td}_{R_m} = \mathbf{Sig}(m)$. On input $(id, c) = (m, f(x))$, Invert knowing $\mathsf{td}_{R_m} = \mathbf{Sig}(m)$ generates an OR-signature π for the knowledge of a pre-image of $f(x)$ or $\mathbf{Sig}(m)$.

Correctness property is obvious and ambiguity property results from the fact that the OR-proof Σ_{OR} is witness indistinguishable. Now, consider the one-wayness property. The attacker A against $\mathcal{R}_{OR}$ gets $D_{\mathcal{R}_{OR}} = pk$ as input and has access to the signing oracle $O_{\mathbf{Sig}}$. A wins if it comes up with m which was not queried to $O_{\mathbf{Sig}}$ such that for a given lock $f(x) \in \mathcal{L}_{R_m}$, A can find an OR-signature π for the knowledge of a pre-image of $f(x)$ or $\mathbf{Sig}(m)$. However, the Fiat-Shamir proof is actually proof of knowledge and the ability to come up with a valid proof implies that we can extract a valid witness which is either a new signature value or a pre-image of the one-way function. Therefore, if A succeeds, we can either forge an ordinary signature or invert the one-way function, both of which easily lead to contradiction to the security of the underlying signature scheme and one-way function. Finally, the soundness property can be checked from the correctness property of the OR-proof Σ_{OR}. □

Remark 1. (Σ-PROTOCOLS) The Σ-protocol Σ_f for the knowledge of a pre-image of a one-way function and $\Sigma_{\mathbf{Sig}}$ for the knowledge of a signature value can be constructed in generic ways [18]. However, there exist very efficient Σ-protocols for specific cases. For example, Σ-protocol in [24] can be used for the RSA function or the FDH signature scheme [5], and Σ-protocol in [34] can be applied to the discrete logarithm function or the BLS signature scheme [8]. While efficient Σ-protocols for the knowledge of a signature value in [24, 34] require the random oracle model, relatively efficient Σ-protocols for the knowledge of a signature value without the random oracle model can be founded in [12, 13, 9].

Remark 2. (ALTERNATIVE TO THE FIAT-SHAMIR PROOF – I) Notice that the proof of Theorem 1 only requires the following properties from the Fiat-Shamir proof: (1) witness indistinguishability and (2) proof of knowledge. Therefore, we can use the straight-line extractable WI proof [28] instead of the Fiat-Shamir proof. Like the Fiat-Shamir proof, the construction of the straight-line extractable WI proof starts with Σ-protocol but the length of the resulting proof is much longer. However, non-programmable random oracle can be used and better exact security is obtained. Therefore, the choice depends on the tradeoff between efficiency and exact security.

Remark 3. (ALTERNATIVE TO THE FIAT-SHAMIR PROOF – II) Instead of the Fiat-Shamir proof, we can also use non-interactive witness indistinguishable proofs of knowledge (for 'I know the pre-image of $f(x)$' or 'I know the signature value $\mathbf{Sig}(m)$'). In this case, we do not need the random oracle and can use instead a common reference string (which can be included in the public key pk). However, the best known way of constructing non-interactive witness indistinguishable proofs of knowledge requires the existence of trapdoor one-way permutations [33] and is extremely inefficient. Nevertheless, this observation leads to the following corollary.

Corollary 1. *If there is a trapdoor one-way permutation, there exists a secure* ID-THIR *in the standard model.*

3 Time Capsule Signature

3.1 Definition

Definition 2. *A* time capsule signature scheme *is specified by an 8-tuple of* PPT *algorithms* $(\mathsf{Setup}^{\mathsf{TS}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{TSig}, \mathsf{TVer}, \mathsf{TRelease}, \mathsf{Hatch}, \mathsf{PreHatch}, \mathsf{Ver})$ *such that:*

- $\mathsf{Setup}^{\mathsf{TS}}$. *This setup algorithm is run by Time Server. It takes a security parameter as input and returns a public/private time release key pair* $(\mathsf{TPK}, \mathsf{TSK})$.
- $\mathsf{Setup}^{\mathsf{User}}$. *This seup algorithm is run by each user. It takes a security parameter as input and returns the user's public/private key pair* $(\mathsf{PK}, \mathsf{SK})$.
- TSig. *The time capsule signature generation algorithm* TSig *takes as input* $(m, \mathsf{SK}, \mathsf{TPK}, t)$, *where* t *is the specific time from which the signature becomes valid. It outputs a time capsule signature* σ'_t.
- TVer. *The time capsule signature verification algorithm* TVer *takes as input* $(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t)$ *and outputs* 1 (*accept*) *or* 0 (*reject*).
- $\mathsf{TRelease}$. *The time release algorithm* $\mathsf{TRelease}$ *is run by Time Server and takes as input* (t, TSK). *At the beginning of each time period* t, *Time Server publishes* $Z_t = \mathsf{TRelease}(t, \mathsf{TSK})$. *Note that Time Server dose not contact any user at any time and need not know anything about the users.*
- Hatch. *This algorithm is run by any party and is used to open a valid time capsule signature which became mature. It takes as input* $(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, Z_t)$ *and returns the hatched signature* σ_t.
- $\mathsf{PreHatch}$. *This algorithm is run by the signer and used to open a valid time capsule signature which is not mature yet. It takes as input* $(m, \sigma'_t, \mathsf{SK}, \mathsf{TPK}, t)$ *and returns the pre-hatched signature* σ_t.
- Ver. *This algorithm is used to verify a hatched* (*or pre-hatched*) *signature.* Ver *takes as input* $(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t)$ *and returns* 1 (*accept*) *or* 0 (*reject*).

Correctness property states that

- $\mathsf{TVer}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{PK}, \mathsf{TPK}, t) = 1$ *and*
- $\mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) = 1$, *where* $\sigma_t = \mathsf{Hatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{PK}, \mathsf{TPK}, Z_t)$ *or* $\sigma_t = \mathsf{PreHatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{SK}, \mathsf{TPK}, t)$.

Ambiguity property states that

– *The "hatched signature"* $\sigma_t = \mathsf{Hatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{PK}, \mathsf{TPK}, Z_t)$ *is computationally indistinguishable from the "pre-hatched signature"* $\sigma_t = \mathsf{PreHatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t),\ \mathsf{SK}, \mathsf{TPK}, t)$*, even if the distinguisher knows* TSK*.*

The security of time capsule signatures consists of ensuring three aspects: security against the signer Alice, security against the verifier Bob, and security against Time Server. In the following, the oracle simulating the time capsule signature generation algorithm TSig is denoted by O_{TSig}, the oracle simulating the time release algorithm $\mathsf{TRelease}$ by O_{TR}, and the oracle simulating $\mathsf{PreHatch}$ by O_{PreH}. The oracle O_{TSig} takes (m, t) as input and returns Alice's time capsule signature σ'_t.[4] The oracle O_{PreH} takes (m, t, σ'_t) as input and returns Alice's pre-hatched signature σ_t.

Security against Alice. We require that any PPT adversary A succeeds with at most negligible probability in the following experiment.

$$\begin{aligned}
\mathsf{Setup}^{\mathsf{TS}}(1^k) &\to (\mathsf{TPK}, \mathsf{TSK})\\
(m, t, \sigma'_t, \mathsf{PK}) &\leftarrow A^{O_{\mathsf{TR}}}(\mathsf{TPK})\\
Z_t &\leftarrow \mathsf{TRelease}(t, \mathsf{TSK})\\
\sigma_t &\leftarrow \mathsf{Hatch}(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, Z_t)\\
\text{success of } A &= [\mathsf{TVer}(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t) \stackrel{?}{=} 1 \ \wedge\ \mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) \stackrel{?}{=} 0]
\end{aligned}$$

In other words, Alice should not be able to produce a time capsule signature σ'_t, where σ'_t looks good to Bob but cannot be hatched into Alice's full signature by the honest Time Server.

Security against Bob. We require that any PPT adversary B succeeds with at most negligible probability in the following experiment.

$$\begin{aligned}
\mathsf{Setup}^{\mathsf{TS}}(1^k) &\to (\mathsf{TPK}, \mathsf{TSK})\\
\mathsf{Setup}^{\mathsf{User}}(1^k) &\to (\mathsf{PK}, \mathsf{SK})\\
(m, t, \sigma_t) &\leftarrow B^{O_{\mathsf{TSig}}, O_{\mathsf{TR}}, O_{\mathsf{PreH}}}(\mathsf{PK}, \mathsf{TPK})\\
\text{success of } B &= [\mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) \stackrel{?}{=} 1 \ \wedge\ t \notin \mathit{Query}(B, O_{\mathsf{TR}})\\
&\qquad \wedge\ (m, t, \cdot) \notin \mathit{Query}(B, O_{\mathsf{PreH}})]
\end{aligned}$$

where $\mathit{Query}(B, O_{\mathsf{TR}})$ is the set of queries B asked to the time release oracle O_{TR}, and $\mathit{Query}(B, O_{\mathsf{PreH}})$ is the set of *valid* queries B asked to the oracle O_{PreH} (i.e., (m, t, σ'_t) such that $\mathsf{TVer}(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t) = 1$). In other words, Bob should not be able to open a pre-mature time capsule signature without help of the singer or Time Server. Notice that Bob can make any time release query to

[4] We assume that the adversary attacks an honest user Alice. The adversary can collude with all other (dishonest) users.

O_{TR} except the target time t. Therefore, the above experiment requires strong security guaranteeing both forward and backward secrecy.

Security against Time Server. We require that any PPT adversary C succeeds with at most negligible probability in the following experiment.

$$\begin{aligned}
&\mathsf{Setup}^{\mathsf{TS}^*}(1^k) \to (\mathsf{TPK}, \mathsf{TSK}^*)\\
&\mathsf{Setup}^{\mathsf{User}}(1^k) \to (\mathsf{PK}, \mathsf{SK})\\
&\quad (m, t, \sigma_t) \leftarrow C^{O_{\mathsf{TSig}}, O_{\mathsf{PreH}}}(\mathsf{PK}, \mathsf{TPK}, \mathsf{TSK}^*)\\
&\text{success of } C = [\mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) \stackrel{?}{=} 1 \ \wedge \ (m, \cdot) \notin Query(C, O_{\mathsf{TSig}})]
\end{aligned}$$

where $\mathsf{Setup}^{\mathsf{TS}^*}$ denotes the run of $\mathsf{Setup}^{\mathsf{TS}}$ with a dishonest Time Server (run by C), TSK^* is C's state after this run, and $Query(C, O_{\mathsf{TSig}})$ is the set of queries C asked to the time capsule signature generation oracle O_{TSig} (i.e., $(m, t') \notin Query(C, O_{\mathsf{TSig}})$ for all t'). In other words, Time Server should not be able to produce a valid hatched or pre-hatched signature on m of Alice without explicitly asking Alice to produce a time capsule signature on m.

3.2 Generic Construction Based on ID-THIR

The scheme. Let $(\mathbf{Set}, \mathbf{Sig}, \mathbf{Ver})$ be an ordinary signature scheme and (Gen, Sample, Check, Extract, Invert) be the procedures for ID-THIR.

- $\mathsf{Setup}^{\mathsf{TS}}$. Time Server chooses $(D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}})$ by running $\mathsf{Gen}(1^k)$ and sets $(\mathsf{TPK}, \mathsf{TSK}) = (D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}})$.
- $\mathsf{Setup}^{\mathsf{User}}$. Each user chooses (pk, sk) by running $\mathbf{Set}(1^k)$ and sets $(\mathsf{PK}, \mathsf{SK}) = (pk, sk)$.
- TSig. To generate a time capsule signature on a message m for time t, the signer gets a random lock/proof pair (c, d) from $\mathsf{Sample}_{D_{\mathcal{R}}}(t)$ and computes $s = \mathbf{Sig}_{sk}(m||c||t)$. The time capsule signature value σ'_t is (s, c) and the signer stores the proof d for later use.
- TVer. For a given time capsule signature $\sigma'_t = (s, c)$, the verifier checks that $c \in \mathcal{L}_{R_t}$ and s is a valid signature on $m||c||t$ by running $\mathbf{Ver}_{pk}(m||c||t, s)$.
- TRelease. For a given time value t, Time Server computes $\mathsf{td}_{R_t} = \mathsf{Extract}_{\mathsf{mtd}_{\mathcal{R}}}(t)$ and publishes $Z_t = \mathsf{td}_{R_t}$.
- Hatch. To open a mature time capsule signature $\sigma'_t = (s, c)$, a party computes $\widetilde{d} = \mathsf{Invert}_{\mathsf{td}_{R_t}}(c)$ and returns the hatched signature $\sigma_t = (s, c, \widetilde{d})$.
- PreHatch. To open a valid pre-mature time capsule signature $\sigma'_t = (s, c)$, the signer returns the pre-hatched signature $\sigma_t = (s, c, d)$ where the proof d is a stored value in the stage of TSig.
- Ver. For a given hatched (or pre-hatched) signature $\sigma_t = (s, c, d)$, the verifier checks the lock/proof pair by running $\mathsf{Check}_{D_{\mathcal{R}},t}(c, d)$. Then, he verifies that s is a valid signature on $m||c||t$ by running $\mathbf{Ver}_{pk}(m||c||t, s)$.

The correctness property and the ambiguity property of the scheme are obvious from the properties of ID-THIR. We now analyze its security.

Theorem 2. *The time capsule signature scheme presented above is secure if the underlying ordinary signature scheme and the* ID-THIR *are secure.*

Proof: We prove the security against Alice, Bob, and Time Server.

Security against Alice. Security against Alice follows unconditionally. A valid time capsule signature $\sigma'_t = (s, c)$ satisfies that $c \in \mathcal{L}_{R_t}$ and $\mathbf{Ver}_{pk}(m||c||t, s) = 1$. If Time Server releases $\mathsf{td}_t = \mathsf{Extract}_{\mathsf{mtd}_\mathcal{R}}(t)$, any party can obtain a proof $\widetilde{d} = \mathsf{Invert}_{\mathsf{td}_t}(c)$ for the lock $c \in \mathcal{L}_{R_t}$. By the correctness property of ID-THIR, $\mathsf{Check}_{D_\mathcal{R},t}(c, \widetilde{d}) = 1$ always holds. Therefore, the hatched signature $\sigma_t = (s, c, \widetilde{d})$ passes the verification algorithm Ver.

Security against Bob. To show security against Bob, we convert any attacker B that attacks our time capsule signature scheme into an inverter Inv of ID-THIR. Recall that Inv gets $D_\mathcal{R}$ as input and has access to the trapdoor extraction oracle O_{Extract}. Inv wins if it comes up with id which was not queried to O_{Extract} s.t. for a given lock $c \in L_{R_{id}}$, Inv can find a proof d for c. On the other hand, B expects $(\mathsf{PK}, \mathsf{TPK})$ as input and has access to $O_{\mathsf{TSig}}, O_{\mathsf{TR}}, O_{\mathsf{PreH}}$. B wins if it forges a hatched (or pre-hatched) signature σ_t of some message m without asking t to O_{TR} or (m, t, σ'_t) to O_{PreH}. Let (m_B, t_B, σ_{t_B}) be the successful forgery of the attacker B. We can assume that B obtained the corresponding time capsule signature σ'_{t_B} from O_{TSig}, since the underlying ordinary signature scheme $(\mathbf{Set}, \mathbf{Sig}, \mathbf{Ver})$ is existentially unforgeable against chosen message attacks.

When Inv receives $D_\mathcal{R}$ from an ID-THIR challenger $\mathcal{C}$, it begins simulating the attack environment of B. Inv picks a random public/private key pair (pk, sk) by running $\mathbf{Set}(1^k)$, sets $\mathsf{PK} = pk$, $\mathsf{SK} = sk$, $\mathsf{TPK} = D_\mathcal{R}$, and gives $(\mathsf{PK}, \mathsf{TPK})$ to B. Inv manages a list $L = \{(m_i, t_i, s_i, c_i, d_i)\}$ to answer B's queries to O_{PreH}. Let q_{TSig} be the total number of O_{TSig} queries made by B and r be a random number chosen by Inv in the interval of $\{1, 2, \cdots, q_{\mathsf{TSig}}\}$. Now, Inv knowing $\mathsf{SK} = sk$ responds to the i-th O_{TSig} query (m_i, t_i) of B as follows;

- If $i = r$, Inv outputs t_r to the challenger $\mathcal{C}$ and receives a random lock $c \in R_{t_r}$ from the challenger. Inv sets $c_r = c$ and computes $s_r = \mathbf{Sig}_{sk}(m_r||c_r||t_r)$. Inv returns $\sigma'_{t_r} = (s_r, c_r)$ to B and stores the element $(m_r, t_r, s_r, c_r, \perp)$ in the list L.
- If $i \neq r$, Inv picks a random lock/proof pair (c_i, d_i) from $\mathsf{Sample}_{D_\mathcal{R}}(t_i)$ and computes $s_i = \mathbf{Sig}_{sk}(m_i||c_i||t_i)$. Inv returns $\sigma'_{t_i} = (s_i, c_i)$ to B and stores the element $(m_i, t_i, s_i, c_i, d_i)$ in the list L.

To simulate O_{TR} to the query t_i of B, Inv simply asks t_i to its own trapdoor extraction oracle O_{Extract} and gets $\mathsf{td}_{R_{t_i}}$. If $t_i = t_r$, Inv abort. Otherwise, Inv returns $Z_{t_i} = \mathsf{td}_{R_{t_i}}$ to B.

To simulate O_{PreH} to the query (m_i, t_i, s_i, c_i), Inv checks whether the query is in the list L or not (by considering only the first four components of an element in L). 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), Inv 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), Inv obtains a proof d_i from the list L and give a pre-hatched signature $\sigma_{t_i} = (s_i, c_i, d_i)$ to

B. If (m_i, t_i, s_i, c_i) is not in the list L (i.e., the time capsule signature was not generated by Inv and therefore the query is invalid with very high probability), Inv answers randomly to B.

The probability that Inv does not abort during the simulation is at least $1/q_{\mathsf{TSig}}$, since $r \in \{1, \cdots, q_{\mathsf{TSig}}\}$ is randomly chosen and a secure ID-THIR satisfies the ambiguity property. When B outputs the forgery $(m_B, t_B, s_B, c_B, d_B)$, Inv verifies that the forgery passes the verification algorithm Ver and $(m_B, t_B, s_B, c_B) = (m_r, t_r, s_r, c_r)$. If so, Inv outputs the proof d_B. Otherwise, Inv chooses a proof d_{Inv} randomly and outputs d_{Inv}. Therefore, if B forges with a probability ϵ, Inv succeeds in breaking the one-wayness of ID-THIR with a probability $\epsilon' \geq \epsilon/q_{\mathsf{TSig}}$.

Security against Time Server. To show security against Time Server, we convert any attacker C that attacks our time capsule signature scheme into a forger F for the underlying ordinary signature. Recall that F gets pk as an input, and has access to the signing oracle $O_{\mathbf{Sig}}$. On the other hand, C expects $(\mathsf{PK}, \mathsf{TPK}, \mathsf{TSK})$ as input and has access to O_{TSig} and O_{PreH}. C wins if it forges a hatched (or pre-hatched) signature σ_t of some message m without obtaining a time capsule signature on m from O_{TSig}.

So here is how F simulates the run of C. To choose ID-THIR, F runs $\mathsf{Gen}(1^k)$ and obtains $(D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}})$. Then, F gives $(\mathsf{PK}, \mathsf{TPK}, \mathsf{TSK}) = (pk, D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}})$ to C. F can respond to O_{TSig} queries (m_i, t_i) of C by choosing a random lock/proof pair (c_i, d_i) from $\mathsf{Sample}_{D_{\mathcal{R}}}(t_i)$ and getting an ordinary signature s_i on $m_i||c_i||t_i$ from its own signing oracle $O_{\mathbf{Sig}}$. F stores (m_i, c_i, d_i, t_i) in the list $L = \{(m_i, c_i, d_i, t_i)\}$ to answer C's queries to O_{PreH}. To simulate O_{PreH} to the queries (m_i, t_i, s_i, c_i), F verifies that s_i is a valid signature on $m_i||c_i||t_i$.

- If s_i is a valid signature on $m_i||c_i||t_i$, F checks whether (m_i, c_i, t_i) is in the list L or not. If it is in the list, F can give the corresponding proof d_i from the list L. Otherwise, s_i is a new signature value and F succeeds in producing a new forgery s_i on $m_i||c_i||t_i$. F stops the simulation.
- If s_i is not a valid signature on $m_i||c_i||t_i$, F answers randomly.

When C outputs the forgery $(\widehat{m}, \widehat{t}, \widehat{\sigma_t})$ where $\widehat{\sigma_t} = (\widehat{s}, \widehat{c}, \widehat{d})$, F outputs an ordinary signature $\widehat{s}$ on a message $\widehat{m}||\widehat{c}||\widehat{t}$. Therefore, if C succeeds with a probability ϵ, F succeeds in producing a new forgery with a probability $\epsilon' \geq \epsilon$. □

Theorem 3. *If there is a one-way function, there exists a secure time capsule signature scheme in the random oracle model.*

Proof: Secure signatures exist if and only if one-way functions exist [32]. Together with Theorem 1 and Theorem 2, we obtain Theorem 3. □

Theorem 4. *If there is a trapdoor one-way permutation, there exists a secure time capsule signature scheme in the standard model.*

Proof: Secure signatures exist if and only if one-way functions exist [32]. Together with Corollary 1 and Theorem 2, we obtain Theorem 4. □

Remark 4. (Event Capsule Signature) In the definition and construction of time capsule signature, we did not use any characteristic of the real time. Actually, t need not be a time value and any index works for t. Therefore, the definition and construction of time capsule signature can be efficiently converted to those of event capsule signature.

4 On Trapdoor Hard-to-Invert Relation

A trapdoor hard-to-invert relation (THIR) is a specific elementary relation R_{id} in ID-THIR $\mathcal{R} = \{R_{id} \mid id \in I_R\}$. The definition of THIR can be derived from that of ID-THIR and the construction becomes even simpler as a signature on one identity is simply a one-way function. Notice that THIR is also very easily constructed without the random oracle model (unlike ID-THIR) if trapdoor one-way permutations exist (for details, refer to Appendix A).

However, it is interesting to ask whether THIR (primitive simpler than ID-THIR) can be built from one-way functions (or even one-way permutations) in the standard model. We leave this as an open problem. However, we comment that it is highly unlikely that a special case of THIR — so called deterministic THIR where only one proof d exists for a given lock c — can be constructed from one-way permutations. Indeed, one can easily see that the existence of a deterministic THIR implies that of a secure key agreement scheme.[5] However, it is known that there exists no block-box reduction from one-way permutations to secure key agreement schemes [25].

Acknowledgments

We would like to thank Leonid Reyzin for several very insightful discussions. The work of the second author was partially supported by the Brain Korea 21 project.

[5] Alice first sends Bob a randomly generated D_R while the corresponding trapdoor td_R is kept secret. After receiving D_R, Bob samples a random lock/proof pair $(c, d) \in R$ and sends c to Alice. By using the trapdoor td_R, Alice can recover (unique) d from c. For secure communication, Alice and Bob can use d as a secret key (or derive a secure key from d).

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In K. Nyberg, editor, *Advances in Cryptology—EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer-Verlag, 31 May–4 June 1998.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communication*, 18(4), pages 593–610, 2000.
3. M. Bellare and S. Goldwasser. Encapsulated key escrow. MIT Laborator for Computer Science Technical Report 688, Apr. 1996.
4. M. Bellare and S. Goldwasser. Verifiable partial key escrow. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 78–91. 1–4 Apr. 1997.
5. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 62–73. 3–5 Nov. 1993.
6. I. Blake and A. Chan. Scalable, server-passive, user-anonymous timed release public key encryption from bilinear pairing. IACR E-print Archive. Available from `http://eprint.iacr.org/2004/211/`, 2004.
7. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 19–23 Aug. 2001.
8. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology—ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 9–13 Dec. 2001.
9. D. Boneh, B. Lynn and H. Shacham. Short Group Signatures. In M. Franklin, editor, *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 15–19 Aug. 2004.
10. D. Boneh and M. Naor. Timed commitments. In *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer-Verlag, 20–24 Aug. 2000.
11. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 4 May–8 May 2003.
12. J. Camenisch and A. Lysyanskaya. Signature schemes with efficient protocols. In S. Cimato, C. Galdi and G. Persiano, editor, *Security in Communication Networks—SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer-Verlag, 11–13 Sep. 2002.
13. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer-Verlag, 15–19 Aug. 2004.
14. D. Catalano, D. Pointcheval, and T. Pornin. IPAKE: isomorphisms for password-based authenticated key exchange. In M. Franklin, editor, *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*. pages 477–493. Springer-Verlag, 15–19 Aug. 2004.

15. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *Advances in Cryptology—CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 21–25 Aug. 1994.
16. Y. Dodis and L. Reyzin. Breaking and repairing optimistic fair exchange from PODC 2003. In *Proceedings of the ACM workshop on Digital Rights Management 2003*, pages 47–54. 27 Oct. 2003.
17. C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. Brickell , editor, *Advances in Cryptology—CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 139–147. 16–20 Aug. 2004.
18. U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in Cryptology—CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer-Verlag, 20–24 Aug. 1989.
19. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 416–426. 14-16 May 1990.
20. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. Odlyzko, editor, *Advances in Cryptology—CRYPTO 1986* volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 11–15 Aug. 1986.
21. J. Garay and M. Jakobsson. Timed release of standard digital signatures. In *Financial Cryptography 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 168–182. Springer-Verlag, 11–14 Mar. 2002.
22. J. Garay and C. Pomerance. Timed fair exchange of standard signatures. In *Financial Cryptography 2003*, volume 2742 of *Lecture Notes in Computer Science*, pages 190–207. Springer-Verlag, 27–30 Jan. 2003.
23. O. Goldreich, S. Micali and A.Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM,* 38(3), pages 691–729, 1991.
24. L. Guillou and J.J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 21–25 Aug, 1988.
25. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61. 15-17 May 1989.
26. A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. IACR E-print Archive. Available from `http://eprint.iacr.org/2001/003/`, 2001.
27. I. Osipkov, Y. Kim and J. Cheon. New approaches to timed-release cryptography IACR E-print Archive. Available from `http://eprint.iacr.org/2004/231/`, 2004.
28. R. Pass. On deniability in the common reference string and random oracle model. In *Advances in Cryptology—CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337. Springer-Verlag, 17–21 Aug. 2003.
29. R. Rivest, A. Shamir, and D. Wagner. Time lock puzzles and timed release cryptography. Technical report, MIT/LCS/TR-684.
30. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2), pages 120–126. ACM, Feb. 1978.

31. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology—EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 12–16 May 1996.
32. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394. 14–16 May 1990.
33. A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 427–436. 24–27 Oct. 1992.
34. C. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology—CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, 20–24 Aug. 1989.
35. A. Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and D. Chaum, editor, *Advances in Cryptology—CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 19–22 Aug. 1984.

Appendix A. Trapdoor Hard-to-Invert Relation

Definition 3. *A* trapdoor hard-to-invert relation (THIR) *is a relation R, where sampling a random lock/proof pair $(c, d) \in R$ is easy but finding a proof for a given lock is difficult without knowing the trapdoor.* THIR *can be specified by a 4-tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Sample}, \mathsf{Check}, \mathsf{Invert})$ *such that:*

- Gen*. This algorithm is used to generate a relation R and the trapdoor information.* $\mathsf{Gen}(1^k)$ *returns D_R (the description of R) and* td_R *(the trapdoor).*
- Sample*. This sampling algorithm takes as input D_R and returns a random lock/proof pair $(c, d) \in R$.*
- Check*. This algorithm is used to check the validity of the proof. If $(c, d) \in R$, then* $\mathsf{Check}_{D_R}(c, d)$ *returns 1 (accept). Otherwise, it returns 0 (reject).*
- Invert*. This algorithm is used to find a proof d for $c \in \mathcal{L}_R$ by using* td_R*. If $c \in \mathcal{L}_R$, then* $\mathsf{Invert}_{\mathsf{td}_R}(c)$ *returns a proof d such that $(c, d) \in R$.*

Let $(c, d) \leftarrow \mathsf{Sample}(D_R)$ *and* $\widetilde{d} \leftarrow \mathsf{Invert}_{\mathsf{td}_R}(c)$*. Correctness property states that* $\mathsf{Check}_{D_R}(c, d) = 1$ *and* $\mathsf{Check}_{D_R}(c, \widetilde{d}) = 1$*, and ambiguity property states that (c, d) and $(c, \widetilde{d})$ are computationally indistinguishable, even if the distinguisher knows* td_R*. One-wayness property states that the following probability is negligible for all PPT algorithm A:*

$$\Pr[\mathsf{Check}_{D_R}(c, \widehat{d}) = 1 \mid (D_R, \mathsf{td}_R) \leftarrow \mathsf{Gen}(1^k); (c, d) \leftarrow \mathsf{Sample}(D_R); \widehat{d} \leftarrow A(D_R, c, 1^k)]$$

Soundness property states that the following probability is negligible for all algorithm B:

$$\Pr[c \in \mathcal{L}_R \wedge \mathsf{Check}_{D_R}(c, \widetilde{d}) = 0 \mid (D_R, \mathsf{td}_R) \leftarrow \mathsf{Gen}(1^k); c \leftarrow B(D_R); \widetilde{d} \leftarrow \mathsf{Invert}_{\mathsf{td}_R}(c)]$$

If THIR *satisfies these four properties, we say that* THIR *is secure.*

Constructions. We first present THIR based on RSA permutation [30] in the standard model. An RSA permutation is defined by $f_{n,e}(x) = x^e \bmod n$, where n is the product of two primes p and q, $x \in \mathbb{Z}_n^*$, $e \in \mathbb{Z}_{\varphi(n)}^*$. Inverting an RSA permutation is believed to be hard (i.e., the RSA assumption). Now, we can define the relation $R_{RSA}^{(n,e)} = \{(y, x) \mid x \in \mathbb{Z}_n^*,\ y = x^e \bmod n\}$ and the algorithms $(\mathsf{Gen}, \mathsf{Sample}, \mathsf{Check}, \mathsf{Invert})$ as follows; Gen chooses primes p, q and an integer $e \in \mathbb{Z}_\varphi(n)$, where $n = pq$. Gen outputs $D_{R_{RSA}} = (n, e)$ and $\mathsf{td}_{R_{RSA}} = e^{-1} \bmod \varphi(n)$. Sample selects a proof $d \in \mathbb{Z}_n^*$, computes a lock $c = f_{n,e}(d)$, and outputs (c, d). For a given lock/proof pair (c, d), Check verifies whether $c = f_{n,e}(d)$ or not. Invert knowing $\mathsf{td}_{R_{RSA}} = e^{-1} \bmod \varphi(n)$ can find a proof d for a given lock $c \in \mathbb{Z}_n^*$ by $d = f_{n,e^{-1}}(c)$. Correctness property and ambiguity property are obvious, because the RSA permutation is deterministic. One-wayness property results from the RSA assumption and soundness property from the fact that $f_{n,e^{-1}}(\cdot)$ is the inverse permutation of $f_{n,e}(\cdot)$ in $\mathbb{Z}_n^*$. Similarly, we can construct THIR based on any trapdoor one-way permutation.

Theorem 5. *If there is a trapdoor one-way permutation, there exists a secure* THIR *in the standard model.*

Proof: (Sketch) We can build a secure THIR based on a trapdoor one-way permutation in the exactly same way as the THIR based on the RSA permutation mentioned above. The security can also be checked from the properties of trapdoor one-way permutations. We leave the details to readers. □

The existence of trapdoor one-way permutations is a sufficient but not a necessary condition for the existence of THIR. We present an example based on GDH groups [26, 8], which does not exploit trapdoor one-way permutations. A prime order group G is called a GDH group if decisional Diffie-Hellman (DDH) problem of G is easy but computational Diffie-Hellman (CDH) problem of G is difficult (i.e., the GDH assumption). DDH problem is the decision problem of the equality $\log_g h \stackrel{?}{=} \log_c d$ for given $g, h, c, d \in G$ and CDH problem is the computation problem of $d = c^{\log_g h}$ for given $g, h, c \in G$. Let G be a GDH group of order p with generators g, h and $\mathcal{V}_{DDH}$ be an efficient algorithm which solves the DDH problem of G. Now, we can define the relation $R_{GDH}^{(g,h)} = \{(c, d) \mid c \in G, d \in G, \log_g h = \log_c d\}$ and the algorithms (Gen, Sample, Check, Invert) as follows; Gen chooses a GDH group G of order p with a generator g and selects an integer $x \in \mathbb{Z}_p$. Gen computes $h = g^x$ and outputs $D_{R_{GDH}} = (g, h, p)$, $\mathsf{td}_{R_{GDH}} = x$. Sample selects an integer $z \in \mathbb{Z}_p$ and outputs a lock/proof pair $(c, d) = (g^z, h^z)$. For given lock/proof pair (c, d), Check verifies whether $\mathcal{V}_{DDH}(g, h, c, d) = 1$ or not. Invert knowing $\mathsf{td}_{R_{GDH}}$ can find a proof d for a given $c \in G$ by $d = c^x$.

Finally, we show that THIR can be constructed from a one-way function in the random oracle model.

Theorem 6. *If there is a one-way function, there exists a secure* THIR *in the random oracle model.*

Proof: (Sketch) Assume that there is a one way function $f : X \to Y$. Let Σ_f be the Σ-protocol for the knowledge of a pre-image of a common input $f(x) \in f(X)$. If we denote by Σ_{OR} the OR-proof for two Σ_f protocols, we can obtain the corresponding OR-signature scheme by applying the Fiat-Shamir heuristic. Now, we define $D_R = (f, f(x_1))$ where $x_1 \in X$ and $R = \{(f(x_2), \pi) \mid x_2 \in X,\ \pi$ is an OR-signature on $f(x_2)$ for the knowledge of a pre-image of either $f(x_1)$ or $f(x_2)\}$. The algorithms (Gen, Sample, Check, Invert) are defined as follows; Gen chooses $x_1 \in X$ and outputs $D_R = (f, f(x_1))$, $\mathsf{td}_R = x_1$. Sample randomly chooses $x_2 \in X$ and generates an OR-signature π on $f(x_2)$ for the knowledge of a pre-image of either $f(x_1)$ or $f(x_2)$. Sample outputs a lock/proof pair $(c, d) = (f(x_2), \pi)$. For a given $(c, d) = (f(x_2), \pi)$, Check verifies whether π is a valid OR-signature on $f(x_2)$ for the knowledge of a pre-image of either $f(x_1)$ or $f(x_2)$. On input $c = f(x_2)$, Invert knowing $\mathsf{td}_R = x_1$ generates an OR-signature π on $f(x_2)$. The security can be checked in a similar way to Theorem 1. □

Remark 5. (Related Work) Trapdoor hard-to-invert isomorphism, i.e., a special case of our general notion, was recently introduced by Catalano et al. [14] to present a generic password-based key exchange construction.