Timed Encryption and Its Application

Shaoquan Jiang

School of Computer Science and Engineering
University of Electronic Science and Technology of China
Email: shaoquan.jiang@gmail.com

Abstract. In this paper, we propose a new notion of timed encryption, in which the encryption is secure within time t while it is totally insecure after some time $T > t$. We are interested in the case where t and T are both polynomial. We propose a concrete construction that is provably secure in the random oracle model. We show that it can be generically (although inefficient) constructed from a timed commitment of Boneh and Naor (CRYPTO'00). Finally, we apply this primitive to construct a deniable secure key exchange protocol, where the deniability and secrecy both hold adaptively and the adversary has the capability of issuing session state reveal attacks and eavesdropping attacks in the non-eraser model. Our protocol is the first to achieve each of the following properties: adaptive deniability admitting eavesdropping attacks and deniability admitting session state reveal attacks in the non-eraser model. Our protocol works in a timing model (inherited from the timed encryption). However, the requirement is rather weak. It essentially asks a user to respond to a ciphertext as soon as possible and hence does not artificially cause any delay. Our usage of timed encryption for the deniability is to use the forceful decryption to obtain the plaintext and hence does not use any random oracle assumption (even if the secrecy proof needs this).

1 Introduction

We propose a new notion of *timed encryption*. Intuitively, a sender can encrypt a message m using a receiver's public key pk such that the latter can normally decrypt using his private key d. It further requires that any one without d can not decrypt the ciphertext within time t while after time T, any person can decrypt it. In this primitive, neither encryption nor decryption needs a trusted agent or needs any interaction. We are interested in the setting where both t and T are polynomial. Practically, t might be a few seconds and T might be is one day. Simply speaking, a timed encryption is a regular public-key encryption, except that it is secure only in time t while it is completely insecure after time $T > t$. It has applications in auction, delayed message broadcast and deniable authentication/key exchange. In auction, it is desired that during the bidding phase, bidders can cast their bids such that no one can read it within time t while it is publicly readable after time $T > t$. This can be realized using a timed encryption with a private key known to nobody. A more interesting setting is deniable authentication where a sender authenticates a message to a receiver while the latter can not prove to a third party the fact of the sender's presence. To achieve this, the receiver can first send a secret key using the sender's timed encryption. The sender then decrypts this key and uses it to authenticate the message back within time t. Since no one except the sender can reply within time t, the authentication is guaranteed. Finally, after time $T > t$, since anybody can decrypt the authentication key and create the authentication tag, the deniability is achieved.

1.1 Related Works

Timed encryption is related to *timed-release* encryption [27], which can be intuitively interpreted as "send a message into the future". This notion is different from ours. Ours requires that within time t the owner of the private key is the unique person that can decrypt the ciphertext while in their notion no one can decrypt during this period. Due to this difference, none of the known

timed-release encryptions can be easily converted to a timed encryption in our notion. Timed-release encryption has two types of approaches.
Time-lock puzzles. In schemes in [27, 29] and the follow-up [25], a sender generates a RSA scheme and uses the factoring to compute 2^t squarings efficiently in order to encrypt a secret while any other person does not have this factoring and hence has to manually repeat 2^t squarings in order to decrypt it, due to which the decryption delay is achieved. Note this is different from a timed encryption since it is the sender (instead of a receiver) that knows the factoring.
Trusted Agents. In this approach, the time-delay is achieved since the decryption requires a secret from trusted agents who will release it after time t. The release can be done interactively or non-interactively. See [7, 17, 10, 13, 11, 12, 28] for references.

Identity-based encryption (IBE) [4] can achieve the delay through a decryption key control, where the identity includes a particular release-time and the decryption secret is available only after this time. This essentially still assumes a trusted agent.

Our timed encryption is different from theirs in that we do not assume any trusted party.

A more related direction is timed-commitment by Boneh and Naor [8], where a committer can commit to a message m and within time t the message m remains confidential while after time T the de-commitment can be opened forcefully. They used the time-lock puzzle based on the sequential nature of RSA modular squaring as in [29] to construct a timed commitment. They also applied this commitment into construct a timed signature. Garay and Jakobsson [20] applied timed commitment techniques to achieve the timed-release property for some standard signatures.

1.2 Contribution

In this paper, we propose a new notion of timed encryption, in which the encryption is secure within time t while it is totally insecure after time $T > t$. We are interested in the case where t and T can be both polynomial. We propose a concrete construction that is provably secure in the random oracle model. We also propose a generic construction from a timed commitment of Boneh and Naor [8]. Finally, we apply this primitive to construct a deniable secure key exchange protocol, where the deniability and secrecy both holds adaptively and the adversary has the capability of issuing session state reveal attacks and eavesdropping attacks. Our protocol is the first to achieve each of the following properties: adaptive deniability admitting eavesdropping attacks and deniability admitting session state reveal attacks both in the non-eraser model. Our protocol works in a timing model (inherited from the timed encryption). However, our timing assumption only requires a user to respond to a message as soon as possible and hence does not artificially cause any communication delay. Pass [9] noticed that deniability in the random oracle model is not trustable. Our deniability proof for the protocol only uses the forceful decryption algorithm of the timed encryption and hence does not depend on a random oracle assumption (even though the secrecy property of the encryption might use it) .

2 Definitions

Notations. For a set S, $x \leftarrow S$ samples x from S randomly; $A|B$ means A concatenating with B. When the context is clear, we also use AB to denote the concatenation of A with B. We use $negl : \mathbb{N} \to \mathbb{R}$ to denote a *negligible* function: for any polynomial $p(x)$, $\lim_{n\to\infty} negl(n)p(n) = 0$. In this paper, we always use κ to denote the security parameter. PPT stands for probabilistic polynomial time. Algorithm A (e.g., encryption or commitment) with input m and randomness r is written as $A(m; r)$. When r is unspecified, simply write it as $A(m)$. $x \leftarrow A(m)$ denotes the random output of A with input m and unspecified randomness.

2.1 Timed Encryption

Timed encryption essentially is a public-key cryptosystem, which, besides the normal encryption and decryption, has a forceful decryption algorithm without a key but using longer time. In this paper, we are interested in the case where both the forceful decryption and normal decryption (with a key) run in a polynomial time. Here we need to be careful about the runtime as it depends on the space model. Practically, an algorithm implemented in the parallel model runs faster than one in a non-parallel model. This problem has been noticed by Boneh and Naor [8] when they define *timed commitment*. They used the *parallel random access machine* (PRAM) model for this purpose, where an adversary can be implemented using *any* polynomial number of parallel processors. Practically, due to the hardware cost, the degree of parallelism is a priori bounded. Hence, it is useful to consider a bounded PRAM. For a fixed polynomial α, we call an adversary with α processors **an adversary in the α-PRAM model** or simply an **α-PRAM adversary**.

Definition 1. *A triple of algorithms (G, E, D) is a public key encryption. It is a* secure (α, t, T)-timed encryption *if the following holds. Let $(e, d) \leftarrow G(1^\kappa)$.*

- Completeness. *A ciphertext generated through the specification can always be correctly decrypted using d. In addition, there exists an algorithm $\mathfrak{T}$ of time T such that $\mathfrak{T}(C) = D_d(C)$ holds for any string C, except for a negligible probability.*
- Secrecy. *Any PPT α-*PRAM *adversary $\mathcal{A}$ only has a negligible advantage in the chosen ciphertext attack below.*
  - **Challenge.** *$\mathcal{A}$ comes up with two messages m_0, m_1 of equal length as a challenge pair. In turn, he will receive $C_b = \mathcal{E}_e(m_b)$ for $b \leftarrow \{0, 1\}$.*
  - **Decryption.** *$\mathcal{A}$ can issue any decryption query $C \neq C_b$ and receive $D_d(C)$. Finally, he outputs a guess bit b' for b. He succeeds if b' is computed within time t from receiving C_b and $b' = b$.*

If (G, E, D) is a secure (α, t, T)-timed encryption for any polynomial α, then we call it a secure (t, T)-timed encryption.

Note that conventionally a CCA2 adversary can issue decryption queries before a challenge query. However, this is unnecessary for timed encryption since the runtime before a challenge is unrestricted (other than PPT) and the attacker thus can run the forceful decryption himself.

2.2 Timed Commitment

Timed commitment is a special commitment whose secrecy is guaranteed only within a given time. It was proposed by Boneh and Naor [8]. Our timed encryption is motivated by this. A timed commitment consists of a committer S and a receiver R and proceeds in three phases.
Commit phase: To commit to a string $w \in \{0, 1\}^n$, S and R execute a protocol `Com` and the final output of R is a commitment c to w.
Open phase: In the open phase, S sends the string w to R. Then, they execute a protocol `DCom`, at the end of which R obtains a proof that w is the committed value.
Forced open phase: If S refuses to execute the *open phase*, there exists an algorithm `F-Open` that takes c as input and, within time T, outputs w and a proof that w is the commitment in c.

In the security definition, Boneh and Naor requires the commitment remains confidential against any polynomial time PRAM adversary. As for the timed encryption, a α-PRAM model is also practical. We relax Boneh and Naor's definition to the following.

Definition 2. *Timed commitment* $\texttt{TC} = (\texttt{TC.Com}, \texttt{TC.DCom}, \texttt{TC.FO})$ *is* (α, t, T)-secure *if it satisfies:*
Completeness: *When R accepts in the commitment phase, his output c must be a valid commitment for some $w \in \{0,1\}^n$ such that* $\texttt{TC.FO}(c) = w$.
Binding: *If* $\texttt{TC.Com}(w) = c$*, then S can not convince R in the decommitment phase that c is a commitment of $w' \neq w$. This holds information theoretically.*
Soundness: *At the end of commitment phase, R is convinced that there exists a forced open algorithm* $\texttt{TC.FO}(c)$ *that outputs the committed string w in time T.*
Privacy: *For any α-*PRAM *adversary $\mathcal{A}$ of time $t < T$, $|\Pr[\mathcal{A}(tr, w) = 1] - \Pr[\mathcal{A}(tr, w') = 1]|$ is negligible, where tr is the transcript in the commitment phase and the probability is over coins of S and R.*

If $\texttt{TC}$ *is (α, t, T)-secure for any polynomial α, then it is a (t, T)-*secure timed commitment*.*

3 Timed Encryption in the Random Oracle Model

In this section, we construct a concrete timed encryption in the random oracle model. The idea is to decompose a secret into many partial secrets. Each part is not long and is deterministically encrypted. With a decryption key, the secret can be decrypted normally. However, without a decryption key, one (including a PRAM adversary) has to spend a considerable amount of time on determining all partial secrets. This prevents a PRAM adversary obtaining the secret in a short time. On the other hand, if afforded more time, any one can forcefully decrypt all partial secrets (and hence the secret).

Construction 1. Let E be a public key encryption with public key e and private key d and $H : \{0,1\}^* \to \{0,1\}^\ell$ is a hash function. κ is a security parameter and $\delta \in \mathbb{Z}$. β is a constant. $\mathcal{K} : \{0,1\}^* \times \{0,1\}^\ell \to \{0,1\}^*$ is a symmetric key encryption, where ℓ is the key length. To encrypt m, take $r_0 \leftarrow \{0,1\}^\kappa$ and $r_i \leftarrow \{0,1\}^{\beta \log \kappa}, i = 1, \cdots, \delta$, and compute $c_i = E_e[r_i; H(c_0 r_0 r_1 \cdots r_i)]$ and $c_0 = \mathcal{K}[m, H(r_0 r_1 \cdots r_\delta)]$. The ciphertext $C = r_0 c_0 \cdots c_\delta$. To decrypt C using d, first decrypt $\{r_i\}_i$ and then check the consistency of $\{c_i\}_i$ with $\{r_i\}_i$ and c_0. If inconsistent, reject; otherwise, decrypt c_0 using secret key $H(r_0 \cdots r_\delta)$. The forceful decryption is to search for r_1 that is consistent with $c_0 r_0 c_1$. After finding r_1, try r_2 from c_2, r_3 from $c_3, \cdots$. If any of r_i's is not found, reject; otherwise, decrypt c_0 using $H(r_0 \cdots r_\delta)$. Denote this scheme by tE-RO.

Remark. One might think to simplify the ciphertext as $r_0 c_0 c_1$ where the secret key for c_0 is $H(r_0 r_1)$. However, this is insecure since an attacker can simply guess r_1 (hence decrypt c_0) with a non-negligible probability $\kappa^{-\beta}$. Another simplification idea is to modify c_i as $c_i = E[r_i; H(c_0 r_0 r_i)]$. This is not good since $\{c_i\}$ can be decrypted in parallel. In our construction, any attacker (disregarding the degree of his parallelism) has to recover $r_1, \cdots, r_\delta$ almost sequentially. This prevents a PRAM adversary recovering $\{r_i\}$ in a short time. On the other hand, if H has a Merkle tree hash structure (see [21] for details), then $\{H(c_0 r_0 \cdots r_i)\}_{i=1}^{\delta}$ with input $\{r_i\}_{i=1}^{\delta}$ can be computed with δ^2 parallel processors in $\log \delta$ steps. Hence, the decryption using d can be implemented in parallel and each processor only needs to compute one encryption and one decryption of E plus $\log \delta$ evaluations of a basic compressor (in the Merkle tree) with a short input. More efficiency analysis is presented after the security theorem.

Theorem 1. *E is* IND-CPA *secure and $\mathcal{K}$ is semantically secure and one-time unforgeable (unforgeable when given access to one valid ciphertext). H is a random oracle. Then, t*E-RO *is a secure $(\sigma, t, \mu\delta k^\beta)$-timed encryption, where μ is the time to compute one hash of length $\kappa + \beta\delta \log \kappa$ and one encryption of E, $\sqrt{\sigma} t \delta^{-1} \kappa^{-\beta/2} \log \kappa \leq 1/3$ and $\delta \geq \log^{2+\epsilon} \kappa$ for some $\epsilon > 0$.*

Proof. *Completeness.* When a ciphertext is generated normally, it can be decrypted correctly since by the completeness of E, $r_1, \cdots, r_\delta$ can be correctly decrypted and hence the secret key for c_0 is uniquely determined and by the completeness of $\mathcal{K}$, the message m is uniquely decrypted. For any string $C = r_0|c_0|\cdots|c_\delta$ (if C is not in this form, it will be invalid and hence normally rejected), $(r_1, \cdots, r_\delta)$ (exist or not) can be unambiguously determined due to verifying $c_i = E[r_i; H(c_0r_0r_1\cdots r_i)]$ and the unique decryption of E. r_i can be either decrypted using d or sequential search over space $\{0,1\}^{\beta\log\kappa}$. Hence, the forceful decryption and the normal decryption using d have an identical result. The cost for forceful decryption is dominated by δ hashes of length $\kappa + \beta\delta\log\kappa$ and δ encryptions of E. So the completeness follows.
Secrecy. We claim that any σ-PRAM adversary $\mathcal{A}$ can only break the secrecy negligibly; otherwise, an adversary $\mathcal{D}$ can use $\mathcal{A}$ to break the semantic security of $\mathcal{K}$. $\mathcal{D}$ does the following. He generates (e, d) for the public key encryption and invokes $\mathcal{A}$ with e.

- random oracle H. Upon any query x, maintain a list L_H and first check whether x is recorded in L_H. If not, take $y \leftarrow \{0,1\}^\ell$ and add (x, y) into L_H. In any case, return y for $(x, y) \in L_H$ to the requester.
- Upon a challenge query m_0, m_1, $\mathcal{D}$ outputs it as his own challenge query. In turn, he will receive $\mathcal{K}[m_b]$ for $b \leftarrow \{0,1\}$, where the secret key is hidden. He then defines $c_0^* = \mathcal{K}[m_b]$ and generates $r_0^*, r_i^* \leftarrow \{0,1\}^{\beta\log\kappa}, c_i^*, i = 1, \cdots, \delta$ normally. Finally, he outputs $C_b^* = r_0^*c_0^*c_1^*\cdots c_\delta^*$ to $\mathcal{A}$.
- Upon any decryption query $C = r_0c_0c_1\cdots c_\delta (\neq C^*)$, $\mathcal{D}$ decrypts normally. That is, he uses d to normally decrypt $r_1, \cdots, r_\delta$ from $c_1\cdots c_\delta$ and then decrypts c_0 normally.

Finally, when $\mathcal{A}$ outputs a guess bit b' for b within time t from his receiving C_b^*, $\mathcal{D}$ does the following: if $r_0^*\cdots r_\delta^*$ was queried to H oracle by $\mathcal{A}$ or $\mathcal{D}$ himself, he aborts with 0/1 randomly (denote this event by Bad); otherwise, it outputs whatever $\mathcal{A}$ does. The view of $\mathcal{A}$ in the simulation of $\mathcal{D}$ is according to the real distribution unless the following events occur: (1) when $\mathcal{A}$ issues a decryption query $C = r_0c_0c_1\cdots c_\delta$ after receiving C_b^*, it happens that $C \neq C_b^*$ but $r_0c_1\cdots c_\delta$ decrypt to $r_0^*r_1^*\cdots r_\delta^*$. In this case, $c_0 \neq c_0^*$ (otherwise, $C = C_b^*$) and the decryption key $H(r_0^*\cdots r_\delta^*)$ is unknown to $\mathcal{D}$ and hence can not answer. However, since $c_0 \neq c_0^*$, a valid c_0 implies a forgery of ciphertext under the key of c_0^*, contradiction to the one-time unforgeability of $\mathcal{K}$. We ignore this forgery event from now on. (2) when $\mathcal{A}$ issues the H-query $(r_0^*\cdots r_\delta^*)$ (i.e., Bad event occurs), the answer should be the unknown key in c_0^* and so $\mathcal{D}$ can not answer. Since right before Bad event, the view of $\mathcal{A}$ is identical to the real world. Hence, the probability of Bad event will not change if the abortion of $\mathcal{D}$ is replaced by a real execution. Thus, $\Pr[\mathbf{Succ}(\mathcal{D})] \geq \Pr[\mathbf{Succ}(\mathcal{A})] - \Pr[\mathsf{Bad}]/2 + negl(\kappa)$. Finally, since $\mathcal{A}$ has a polynomial time and polynomial parallel processors, so does $\mathcal{D}$. Hence, to prove the theorem, it is sufficient to prove that $\Pr[\mathsf{Bad}]$ is negligible.

Lemma 1. $\Pr[\mathsf{Bad}]$ *is negligible.*

Proof. We prove this lemma for a relaxed adversary $\mathcal{A}$: $\mathcal{A}$ is *any* PPT adversary, except that after receiving C_b^*, he can only issue parallel H-queries for t times and each time includes at most σ H-queries. Here we implicitly assume one hash cost at least one time unit. We now show that the decryption query is unnecessary. To see this, we revise $\mathcal{D}$ such that it does not issue a H-query except in creating C_b^* and the decryption query can be answered only using the existing records for H-queries from $\mathcal{A}$. Let $C = r_0c_0c_1\cdots c_\delta$ be a decryption query. $\mathcal{D}$ acts as follows.

- If $r_0c_0 \neq r_0^*c_0^*$, check whether $r_0c_0r_1'$ for some r_1' was queried to H by $\mathcal{A}$ such that $c_1^* = E[r_1'; H(r_0c_0r_1')]$. If yes, define $r_1 = r_1'$ and continue similarly to determine $r_2, \cdots, r_\delta$; otherwise, reject (this decision is correct by Lemma 2 (Appendix B) since $H(c_0r_0r_1)$ is undefined (thus random)). When $r_1 \cdots r_\delta$ are all determined, check whether $r_0 \cdots r_\delta$ was queried to H. If yes, use $H(r_0 \cdots r_\delta)$ to decrypt c_0; otherwise, reject (this decision is valid due to the unforgeability of $\mathcal{K}$). As a summary, $\mathcal{D}$ can decrypt C based on the records for H-queries from $\mathcal{A}$ only.

- If $r_0 c_0 = r_0^* c_0^*$, $\mathcal{D}$ searches for $r_0^* r_1 \cdots r_\delta$ over all records of H-queries such that it is consistent with $c_1 \cdots c_\delta$. Note such a result is unique if any. If it is found, decrypt c_0 using the record $H(r_0^* r_1 \cdots r_\delta)$; otherwise, reject (this decision is correct due to the unforgeability of $\mathcal{K}$ under the undefined (thus random) key $H(r_0^* r_1 \cdots r_\delta)$). Again, the decryption is answered only based on records for H-queries from $\mathcal{A}$.

Therefore, to prove the lemma, it suffices to prove that any PPT adversary $\mathcal{A}$ that receives C_b^* and issues at most t times of parallel H-queries, can come up with H-query $r_0^* \cdots r_\delta^*$ (i.e., Bad event) negligibly.

Use DisOrd to denote the event: there exists (i, j) with $j \geq i + \log \kappa - 1$ such that the first H-query with prefix $r_0^* \cdots r_i^*$ is a H-query with prefix $r_0^* r_1^* \cdots r_j^*$. We first prove the following claim.

Claim 1. $\Pr[\mathsf{DisOrd}] = negl(\kappa)$ holds for any PPT $\mathcal{A}$ in the PRAM model (not just σ-PRAM model).

Proof. Otherwise, since there are at most δ^2 possible (i, j) pairs, there must exist (i^*, j^*) so that DisOrd on (i^*, j^*) (or $\mathsf{DisOrd}(i^*, j^*)$ for short) occurs non-negligibly. We show that the semantic security of (E, D) for the multiple ciphertexts can be violated by an adversary $\mathcal{B}$. Note semantic security in the multi ciphertexts and that for the single ciphertext are equivalent (e.g., see [21]). $\mathcal{B}$ takes $r_0^*, r_1^*, \cdots, r_\delta^*, m_0, m_1$ and sets the challenge pair for a multi-encryption as $(r_{i^*}^*, \cdots, r_\delta^*)$ or $(0, 0, \cdots, 0)$. In turn, he will receive $\alpha_{i^*}, \cdots, \alpha_\delta$ which is the encryption of either the first vector (corresponding to challenge bit $u = 1$) or all zeros (corresponding to challenge bit $u = 0$). Upon this, he takes $K \leftarrow \{0, 1\}^\ell$, defines $H(r_0^* \cdots r_\delta^*) = K$ and computes $c_0^* = \mathcal{K}(m_b, K)$. He also defines $H(c_0^* r_0^* \cdots r_{i^*}^*)$ as the hidden randomness of α_i for $i \geq i^*$; for $i < i^*$, he computes $c_i^* = E[r_i; H(c_0^* r_0^* r_1^* \cdots r_i^*)]$, where $\mathcal{B}$ maintains H oracle to compute $H(x)$. Finally, he gives $r_0^*, c_0^* \cdots, c_{i^*-1}^*, \{\alpha_j\}_{j=i^*}^\kappa$ to $\mathcal{A}$ and answers the H-query from the latter normally unless a H-query q with prefix $c_0^* r_0^* \cdots r_{i^*}^*$ occurs, which has two cases: if q in fact has a longer prefix $c_0^* r_0^* \cdots r_{j^*}^*$, then $\mathsf{DisOrd}(i^*, j^*)$ occurs; otherwise, $\mathsf{DisOrd}(i^*, j^*)$ will not occur. In the former case, $\mathcal{B}$ terminates with output $u' = 1$; otherwise, outputs $u' = 0$. Note right before query q, the view of $\mathcal{A}$ for the case $u = 1$, is according to the real distribution and hence $\Pr[\mathcal{B}(\{\alpha_i\}_i) = 1] = \Pr[\mathsf{DisOrd}(i^*, j^*)]$; when $u = 0$, the challenge tuple is

$$\langle r_0^*, E[r_1^*; H(r_0^* r_1^*)], \cdots, E[r_{i^*-1}^*; H(r_0^* r_1^* \cdots r_{i^*-1}^*)], E[0], \cdots, E[0] \rangle.$$

We show that $\mathsf{DisOrd}(i^*, j^*)$ in this case occurs negligibly. We show this by reducing to Lemma 5 (Appendix B). In fact, adversary $\mathcal{C}$ in Lemma 5 can carry out the simulation of $\mathcal{B}$ for case $u = 0$ as follows. He prepares $r_0^*, \cdots, r_{i^*-1}^*, c_0^*$ as done by $\mathcal{B}$, lets $r_{i^*-1+v}^* = x_v^*$(unknown), $v = 1, \cdots, \delta - i^* + 1$ and defines $H(c_0^* r_0^* \cdots r_i^*), i = i^*, \cdots, \delta$ randomly without specifying its input. In addition, $n = j^* - i^*$. Finally, H-query $r_0 \cdots r_i$ is answered as follows. When $i < i^*$, the answer is normal as $r_0^* \cdots r_{i^*-1}$ is known; when $r_0 \cdots r_{i^*-1} \neq r_0^* \cdots r_{i^*}^*$ is answered with a random hash result; otherwise, query whether $r_{i^*} \cdots r_i$ equals $x_1^* \cdots x_{i-i^*+1}$ (i.e., $r_{i^*} \cdots r_i$). In case of no, answer the hash result as a random number; in case of yes, Rev_N (in Lemma 5) occurs if and only if $\mathsf{DisOrd}(i^*, j^*)$ in $\mathcal{C}$'s simulation occurs. Hence, when $\mathcal{A}$ makes N times parallel queries, $\Pr[\mathsf{DisOrd}(i^*, j^*)] = \Pr[\mathsf{Rev}_N] \leq \frac{N \cdot Poly(\kappa)}{\kappa^{(j^*-i^*+1)\beta}} \leq \frac{N \cdot Poly(\kappa)}{2^{\beta \log^2 \kappa}}$, negligible, where $Poly(\kappa)$ is the degree of parallelism of $\mathcal{A}$. As $\Pr[\mathsf{DisOrd}(i^*, j^*)]$ is non-negligible for the case $u = 1$, the advantage of $\mathcal{B}$ is non-negligible, contradicting the semantic security of E. □

In the following, we assume DisOrd never occurs. Thus, for any i and $j \geq i + \log \kappa - 1$, the H-query with prefix $r_0^* \cdots r_i^*$ (but not prefix $r_0^* \cdots r_j^*$) always occurs prior to one with prefix $r_0^* \cdots r_j^*$.

Let ν be the number of times that $\mathcal{A}$ makes parallel H-queries. Let n_u denote the number of times $\mathcal{A}$ make parallel H-queries after a query with prefix $r_0^* r_1^* r_2^* \cdots r_{1+(u-1)\log \kappa}^*$ but till a query with prefix $r_0^* r_1^* r_2^* \cdots r_{1+u \log \kappa}^*$ (By Claim 1, $n_u \geq 1$). We have that $\Pr[\nu \leq t] \leq \Pr[\sum_{u=0}^{\delta/\log \kappa - 1} n_u \leq t]$. To bound this probability, we first show the following.

Claim 2. $\Pr[n_0 \leq a_0, \cdots, n_{\delta/\log \kappa - 1} \leq a_{\delta/\log \kappa - 1}] \leq (\sigma/\kappa^\beta)^{\delta/\log \kappa} \cdot \prod_{u=0}^{\delta/\log \kappa - 1} a_u \leq (\frac{t\sigma \log \kappa}{\delta \kappa^\beta})^{\delta/\log \kappa}$.
Proof. Similar to the treatment for DisOrd, we can show that $\Pr[n_0 \leq a_0] \leq \frac{a_0 \sigma}{\kappa^\beta}$. By Lemma 4 (Appendix B), given the view (denoted by y_0) of $\mathcal{A}$ till the first query q_0 with prefix $r_0^* r_1^*$, string $r_{1+\log \kappa} \cdots r_\delta$ remains uniformly random to him, where $x_i^* = r_i^*$ and Ω is the set of queries prior to query q_0 and u_0 is query q_0 ($x_0^* x_1^* x_2 \cdots x_{\log \kappa} \neq x_0^* x_1^* x_2^* \cdots x_{\log \kappa}^*$ is guaranteed by Claim 1). Hence, we can use the approach for bounding $\Pr[\mathsf{DisOrd}]$ to show that $\Pr[n_1 \leq a_1 | y_0] \leq \frac{a_1 \sigma}{\kappa^\beta}$, where y_0 is the adversary view till the query with prefix $r_0^* r_1^*$ occurs. Since DisOrd is assumed not to occur, this inequality holds for all y_0. Since n_0 is deterministic in y_0, by Lemma 3 (Appendix B), $\Pr[n_1 \leq a_1 | n_0 \leq a_0] \leq \frac{a_1 \sigma}{\kappa^\beta}$. Similarly, we can show that $\Pr[n_i \leq a_i | n_0 \leq a_0, n_1 \leq a_1, \cdots, n_{i-1} \leq a_{i-1}] \leq \frac{a_i \sigma}{\kappa^\beta}$. The first inequality follows. The second one is from the geometry inequality. □

We come back to bound $\Pr[\nu \leq t] \leq \Pr[\sum_{u=0}^{\delta/\log \kappa - 1} n_u \leq t]$. Since $n_i \geq 1$, if $t < \delta/\log \kappa$, $\Pr[\nu \leq t] = 0$. If $t \geq \delta/\log \kappa$, since $n_0 + \cdots + n_{\delta/\log \kappa - 1} \leq t$ has $\binom{t}{\delta/\log \kappa} \leq (\frac{3t \log \kappa}{\delta})^{\delta/\log \kappa}$ solutions, by Claim 2, we have that $\Pr[\sum_{u=0}^{\delta/\log \kappa - 1} n_u \leq t] \leq (\frac{\sqrt{3}t\sqrt{\sigma} \log \kappa}{\delta \kappa^{\beta/2}})^{2\delta/\log \kappa} < 3^{-\delta/\log \kappa}$, negligible. Hence, $\Pr[\mathsf{Bad}] = negl(\kappa)$. ■

Corollary 1. $t\mathsf{E\text{-}RO}$ *is a secure* $(\delta/\log \kappa - 1, \mu\kappa^\beta\delta)$*-timed encryption for any* $\delta > \log \kappa$.

Proof. σ-PRAM condition in the proof of Theorem 1 is used only in Claim 2. If $t < \delta/\log \kappa$, Claim 2 is not used in the proof and $\Pr[\nu \leq t] = 0$ (because $n_i \geq 1$), where some negligible events are ignored. ■

Efficiency. $t\mathsf{E\text{-}RO}$ can be setup according to the adversary type. For σ-PRAM adversary, we can set $\beta = \log(t^2\sigma)$ and $\delta = \log^{2.5} \kappa$, where t can be set according to the overall secrecy requirement. Under this setting, the encryption of E^* is dominated by $\log^{2.5} \kappa$ encryptions of E and hashing of H; the normal decryption is dominated by $\log^{2.5} \kappa$ encryptions and decryptions of E and hashing of H. Forceful decryption is dominated by $\frac{1}{2}\kappa^\beta \log^{2.5} \kappa$ encryptions of E and hashing of H. If E uses a hybrid encryption, the cost for normal encryption/decryption of E^* does not increase much with β since the added cost are only from the hashing H and the (symmetric) data encryption part in E, both of which are cheap. When necessary, $E_e^*()/D_d^*()$ can also be implemented in the parallel model for a speedup. t can be taken as large as $\kappa^{\beta/2}/\sqrt{\sigma}$. Hence, the secrecy level can be conveniently set without affecting the normal encryption/decryptions efficiency. For PRAM adversary, the degree of parallelism can be any polynomial. In this setting, we are only guaranteed that the secrecy holds within time $\delta/\log \kappa - 1$ where the time unit is one hash of input size $\kappa + \beta$. In this case, the scheme is meaningful only if $E_e^*()/D_d^*()$ is implemented in the parallel model since otherwise the adversary is only given an attack time that is less than the time for a normal encryption/decryption, which is unreasonable. As said before if H uses Merkle tree hash structure, then $E_e^*()/D_d^*()$ can be implemented in δ^2 parallel-processor model. Each processor has time independent of δ (specifically, one encryption and decryption of E and $\log \delta$ basic compression of small input). It should be noted that this parallel implementation is practical only if δ is small (hence a small secrecy time bound t). In application of deniable key exchange at Section 5, the encryption secrecy is required only to hold for a few seconds. Hence this type of setup is enough. For applications requiring a large δ, our scheme is not suitable.

4 Timed Encryption without a Random Oracle

In the last section, we have constructed a timed encryption in the random oracle model. In the following, we show that one can generically convert a timed commitment into a timed encryption. Since Boneh and Naor [8] constructed a timed commitment without a random oracle, a random oracle free timed encryption exists too. The idea is as follows. A timed commitment already has a forceful opening. But it lacks an efficient way (using a secret) to decrypt. Hence, naturally we can use a normal encryption to encrypt m and use a timed commitment to commit. With a decryption key, one can obtain m from the normal encryption while, without a decryption key, one can forcefully compute m from the timed commitment. To make sure the force commitments and normal encryptions are consistent in m, a non-interactive zero-knowledge (NIZK) proof is used. The formal description follows.

Construction 2. Let (e, d) be a public/private key pair for a public key encryption (E, D). $\texttt{TCom}$ is a timed commitment. $\mathcal{P}$ is a non-interactive zero knowledge using a common random string σ for relation $R_e = \Big\{\langle(E_e[m;r], \texttt{TCom}[m;r']), (m, r, r')\rangle \mid m, r, r' \in \{0,1\}^*\Big\}$. To encrypt m, compute $C = E_e[m;r], \tau = \texttt{TCom}[m;r']$,where r and r' are the randomness for C and τ respectively. $\pi = \mathcal{P}_\sigma[C, \tau; m, r, r']$, where the input is (C, τ) and witness is (m, r, r'). The final ciphertext is $\gamma = (C, \tau, \pi)$. Upon $\gamma = (C, \tau, \pi)$, the normal decryption with d is to first verify if π is valid. If yes, decrypt $m = D_d(C)$; otherwise, $\perp$. Forced decryption algorithm $\mathfrak{T}$ for γ is to first verify π. If valid, forcefully open τ; reject otherwise. Denote this scheme by (E^*, D^*).

Theorem 2. *If (E, D) is CCA2 secure,* $\texttt{TCom}$ *is a secure (α, t, T)-timed commitment and $\mathcal{P}$ is a one-time simulation sound adaptive* NIZK *(See Appendix A), then (E^*, D^*) is a secure (α, t, T)-timed encryption.*

The proof follows from the intuition presented before; see Appendix C for details.

Corollary 2. *In Theorem 2, if* $\texttt{TCom}$ *is a secure (t, T)-timed commitment, then (E^*, D^*) is a secure (t, T)-timed encryption.*

Proof. A secure (t, T)-timed commitment is a secure (α, t, T)-timed commitment for any polynomial α. By Theorem 2, (E^*, D^*) is also a secure (α, t, T)-timed encryption. ■

5 Application to Adaptive Deniable Key Exchange

Deniable key exchange is a protocol that allows two parties to securely establish a common secret while neither of them can prove to a third party that the protocol execution between them has actually occurred. Technically, this can be achieved by requiring that the interaction be simulatable by each of them alone. This property prevents the communication record from being maliciously used as an evidence (i.e., at the court) against an honest user. Deniable key exchange has recently been actively studied in the literature [26, 16, 22]. We are especially interested in the deniability advocated by Di Raimondo et al. [16], where the deniability remains valid even if the adversary can eavesdrop some communication records between honest users. Eavesdropping could add to the difficulty of deniability. Indeed, the eavesdropped transcript is usually linkable to an honest user and its randomness is unknown to the attacker. If an adversary uses part of this transcript in an attack session, this linkage might remain. Adaptivity for deniability is also important, where an adversary can corrupt any user with the time going on. To our knowledge, no adaptive deniable key exchange in the eavesdropping model has been proposed before. In this section, we will resolve this question using a timed encryption (hence in the timing model). Our timing assumption is rather weak. It essentially asks a user to process the ciphertext as soon as possible and hence does not artificially cause any communication delay (unlike [19]). Our deniable security is proven in the non-eraser model, where the intermediate data in the protocol execution can not be erased and, when an instance is corrupted, it has to handled out the state faithfully.

The security model considered here is from Bellare-Rogaway [2] for session key security and from [19, 22] for deniability. Assume there are n parties $P_1, \cdots, P_n$. P_i and P_j might jointly execute a key exchange protocol Ξ to establish a common secret (called *session key*).

Notions. $\Pi_i^{l_i}$ denotes a *protocol instance* in P_i, which is a copy of Ξ running in it and l_i is its instance id. $\mathsf{sid}_i^{l_i}$ is a *session identifier* for $\Pi_i^{l_i}$ and will be specified when analyzing the protocol security. Supposedly, two communicating instances should share the same session identifier. $\mathsf{pid}_i^{l_i}$ is the *partner party* of $\Pi_i^{l_i}$ that he presumably interacts with. $\mathsf{stat}_i^{l_i}$ is the internal state of $\Pi_i^{l_i}$. We also use stat_i to denote an internal state for an unspecified instance in P_i. $sk_i^{l_i}$ is the session key in $\Pi_i^{l_i}$. $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ are partnered if (1) $\mathsf{pid}_i^{l_i} = P_j$ and $\mathsf{pid}_j^{l_j} = P_i$; (2) $\mathsf{sid}_i^{l_i} = \mathsf{sid}_j^{l_j}$. Intuitively, two instances are partnered if they are jointly executing Ξ.

Adversarial Model. Now we introduce the attack model. Essentially, we would like to capture the concern that the adversary can fully control the network. In particular, he can inject, modify, block and delete messages at will. He can also corrupt some users and obtain their secret keys and internal states. He is also able to collect some selected session keys. Finally, Ξ is secure if the session key of any adversarially chosen instance remains computationally random, where the adversary is assumed not to compromise this session key in an obvious way (e.g., corruption or session key request). The formal model is defined as a game between a challenger and an attacker $\mathcal{A}$. The challenger maintains a set of oracles that represent events during protocol executions. Adversarial capabilities are modeled as queries to these oracles adaptively.

$\mathtt{Send}(i, l_i, M)$. In this query, $\mathcal{A}$ can send any message M to $\Pi_i^{\ell_i}$. The result is whatever the latter returns according to the specification. This models P_i's response to an incoming message.

$\mathtt{Reveal}(i, l_i)$. In this query, $\mathcal{A}$ can ask for a session key $sk_i^{l_i}$ in $\Pi_i^{l_i}$. It models a session key loss attack.

$\mathtt{Corrupt}(i, l_i)$. In this query, $\mathcal{A}$ can ask to corrupt $\Pi_i^{\ell_i}$. In turn, state $\mathsf{stat}_i^{\ell_i}$ is given. Note that P_i's long term secret key is not part of $\mathsf{stat}_i^{\ell_i}$. This threat is also called *session state reveal attack* [1, 6]. Security against such an attack essentially means that compromising one session does not affect other sessions.

$\mathtt{Corrupt}(i)$. In this query, $\mathcal{A}$ can corrupt P_i and obtains his long term secret and all internal states $\{\mathsf{stat}_i^{l_i}\}_{l_i}$. In addition, P_i's future action is taken by $\mathcal{A}$. This models the case where some users become malicious.

$\mathtt{Test}(i, l_i)$. This is the security test and is allowed to query it only once. The queried session must be completed and accepted. Furthermore, this session should not be *compromised* (see below for the definition). When this oracle is called, it flips a fair coin b and provides a number α_b to adversary, where $\alpha_0 = sk_i^{l_i}$ and $\alpha_1 \leftarrow \mathcal{K}$ and $\mathcal{K}$ is the space of $sk_i^{l_i}$. The adversary then tries to output a guess bit b'. He is informed of $\mathsf{success}$ if $b' = b$; otherwise, fail.

$\Pi_i^{\ell_i}$ is said $\mathsf{compromised}$ if $\mathtt{Reveal}$ or $\mathtt{Corrupt}$ query was issued to it or its partnered session, or if P_i or $\mathsf{pid}_i^{\ell_i}$ is $\mathtt{Corrupted}$.

The security of the protocol is defined through correctness, secrecy, authentication and deniability.

Correctness. If two partnered instances $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ successfully complete, then $sk_i^{l_i} = sk_j^{l_j}$.
Secrecy. Let $\mathsf{Succ}(\mathcal{A})$ denote the success event in the `Test` query. The secrecy requires that $\Pr[\mathsf{Succ}(\mathcal{A})] < \frac{1}{2} + negl(\kappa)$ (i.e., randomly guessing b is the best strategy).
Authentication. Essentially, authentication is to require that when one instance $\Pi_i^{l_i}$ successfully completes the execution of Ξ, indeed $\mathsf{pid}_i^{l_i}$ attended this execution. Formally, let $\Pi_i^{l_i}$ be the test session and $\mathsf{Non\text{-}Auth}$ be the event: either there does not exist any partnered instance for $\Pi_i^{l_i}$ or its partnered instance is not unique. Then Ξ is said to be *authenticated* if $\Pr[\mathsf{Non\text{-}Auth}(\mathcal{A})]$ is negligible. Note that as mentioned in [22], defining $\mathsf{Non\text{-}Auth}$ on the test session is for simplicity only.

Deniability. Deniability essentially states that the adversary view in the interaction can be simulated himself (i.e., using his knowledge only) and hence its view can not be used as evidences against deniability of other users. Formally, the simulated interaction and the real interaction should be *statistically close*. In our case, it suffices that the oracles can be maintained using all users' public keys and (adaptively) corrupted users' secret keys (when corruption occurs), where a corrupted user's secret key will be provided to the simulator when adversary issues this corruption query. Hence, we can define the deniability as the statistical closeness between the adversary view when the oracles are maintained as specified and the adversary view when the oracles are simulated. As mentioned in the beginning, the deniability should hold against eavesdropping attack. This can be modeled by one more oracle below that is maintained by a trusted party.

- `Execute`(i, l_i, j, l_j). This oracle is maintained by a trusted party. When this oracle is called, a complete protocol execution between $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ is carried out. Finally, $\mathcal{A}$ and the simulator will be provided with a protocol transcript tr. This captures an eavesdropping attack, where $\mathcal{A}$ (and the simulator) does not know the randomness for data in the transcript.

Definition 3. *A key exchange protocol Ξ is* deniable secure *against an adversary class $\mathcal{C}$ if for any $\mathcal{A} \in \mathcal{C}$, correctness, secrecy, authentication and deniability are all satisfied.*

5.1 Construction

In the following, we use a timed encryption to construct a new key exchange protocol that is adaptively deniable secure, where eavesdropping attacks and session state reveal attacks are both allowed. Our protocol is presented as follows (also see Fig. 1).

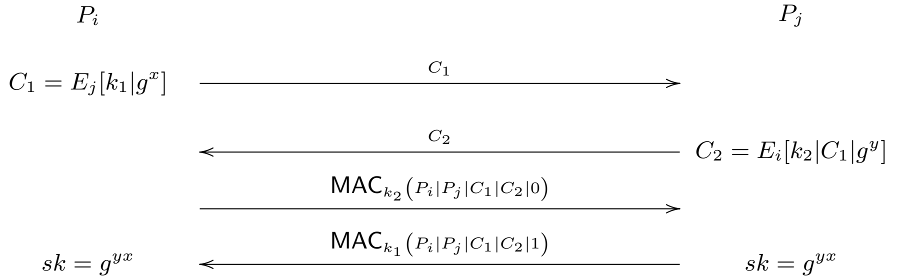

Fig. 1. Our Timed Encryption-based Deniable Key Exchange tE-DKE
(See details in the bodytext)

Initially, take $(E_i, D_i) \leftarrow G(1^\kappa)$ as user P_i's public/private key pair. p, q are large primes with $q \mid p-1$. $g \in \mathbb{Z}_p^*$ has an order of q. MAC: $\{0,1\}^\kappa \times \{0,1\}^* \rightarrow \{0,1\}^\kappa$ is a message authentication code with key space $\{0,1\}^\kappa$. When P_i and P_j want to establish a session key, they proceed as follows.

1. P_i takes $x \leftarrow \mathbb{Z}_q, k_1 \leftarrow \{0,1\}^\kappa$ and sends $C_1 = E_j[k_1|g^x]$ to P_j.
2. Upon C_1, P_j takes $y \leftarrow \mathbb{Z}_q, k_2 \leftarrow \{0,1\}^\kappa$ and sends $C_2 = E_i[k_2|C_1|g^y]$ to P_i.
3. Upon C_2, P_i checks whether $D_i(C_2) = k_2|C_1|Y$ for some $k_2 \in \{0,1\}^\kappa$ and $Y \in \langle g \rangle$. If no, reject; otherwise, send $\mathsf{MAC}_{k_2}(P_i|P_j|C_1|C_2|0)$ to P_j.
4. Upon τ, if $\tau = \mathsf{MAC}_{k_2}(P_i|P_j|C_1|C_2|0)$ and τ is received within time t from sending out round two message and if $D_j(C_1) = k_1|X$ for some $k_1 \in \{0,1\}^\kappa, X \in \langle g \rangle$, set $sk = X^y$ and send $\mathsf{MAC}_{k_1}(P_i|P_j|C_1|C_2|1)$ to P_i; otherwise, reject.
5. Upon σ, If $\sigma \neq \mathsf{MAC}_{k_1}(P_i|P_j|C_1|C_2|1)$ or σ is received more than time t from sending out round one message, reject; otherwise, set $sk = Y^x$.

Remark. To better understand our protocol, some remarks are necessary.

(1) One careful reader might realize that the final message flow seemingly can be moved to the second round (i.e., put together with C_2). Toward this, P_j needs to first derive k_1 from C_1 in order to compute $\mathsf{MAC}_{k_1}(*)$. We show that in this case an adversary can launch a session state reveal attack. The attacker can first sends C_1 to P_j. When P_j sends out C_1 and σ, he reveals the state of the current instance in P_j and obtains k_1. With k_1, the attacker can forge a new C_2' with a known y' and σ'. So when P_i successfully completes, it has no partnered instance in P_j and the session key of this instance is known to the attacker. A simple counter measure to this attack is to erase k_1 after computing σ (noticing k_1 will not be used by P_j anymore). However, this works only in the eraser model, which is not the interest of this work. If τ is sent in the 4th flow as we do, this attack will not occur. Intuitively, this is true since k_1 or k_2 is decrypted only after C_1 and C_2 are fixed and hence the adversary can not choose his own X or Y; otherwise, he can not pass the verification of authentication. Formal treatment is in the security proof.

(2) If C_1 is not encrypted in C_2, the protocol again will suffer from a session state reveal attack. Indeed, when an attacker $\mathcal{A}$ sees P_i's message $C_1 = E_j[k_1|g^x]$, he changes it to $C_1' = E_j[k_1'|g^{x'}]$ and sends it to P_j. After seeing C_2, he forwards to P_i. When P_i sends out τ, $\mathcal{A}$ corrupts the current instance of P_i and obtains k_2, with which $\mathcal{A}$ computes τ' that matches C_1' and C_2. P_j then will be deceptively convinced. In our protocol, C_1 is encrypted in C_2. So $\mathcal{A}$ can not mall C_2 to C_2' that contains C_1' but does not change k_2. Hence, the above attack will fail.

(3) If g^x is not encrypted in C_1 but it is included in the input for τ and σ, then it still suffers from a session state reveal attack. The procedure is similar to the case in item (2).

(4) Deniability of our protocol is intuitively showed as follows. We are required to simulate the oracle response without using an honest user's decryption key. This is done due to the forceful decryption for the timed encryption. Specifically, when receiving C_2, the simulator can freeze the adversary and forcefully decrypt it. Here freezing an adversary intuitively means that the time gap between the occurrence of the communication and the time for the adversary to claim undeniability to an authority is longer than the time to forcefully decrypt a ciphertext. Under this assumption, an adversary's evidence is not convincing since whenever the decryption is needed, he can forcefully decrypt it himself. In timed encryption, the forceful decryption runs in a polynomial time and hence it lies in the complxity class of an adversary. The freezing based simulation is used by Dwork, Naor and Sahai [19].

5.2 Security

In this section, we analyze the security of our protocol. We will show that it satisfies adaptive secrecy, adaptive deniability and authentication. Our adaptive deniability holds even if the adversary can launch an eavesdropping attack. This is the first protocol in the literature that achieves this. Our secrecy property holds even if the adversary can launch a session state reveal attack in the non-eraser model, where all the intermediate data of an instance will not erased and (when revealed) will be given to the adversary. Avoiding such an attack is meaningful as it essentially means that compromising one session should not have a security effect on other non-partnered sessions. Before proceeding, we define the session identifier. For an initiator instance $\Pi_i^{\ell_i}$, let $\mathsf{sid}_i^{\ell_i} = \langle P_i, P_j, E_j[k_1|g^x], C_2\rangle$. Similarly, we define one for a responding instance.

Deniability intuitively follows from the forceful decryption of timed encryption and freezing strategy of [19] as seen in the above remark. The secrecy is to say that an adversary in the secrecy test can not have a success probability better than a näive guess. A routine proof strategy is to use a sequence of game method to modify the oracles gradually such that the final game is easy to analyze. But one pitfall here is that since E is a timed encryption, we can not modify the encryption content; otherwise, the adversary will detect the modification as T is also a polynomial. In our proof, we will only modify the test instance and in addition the simulation will end when time t is elapsed since the modified ciphertext is sent. Hence, the adversary will not have enough time to detect the modification. The authentication property will be showed when we prove the secrecy property, where a test session provably has a unique partner instance.

Theorem 3. *If (G, E, D) is a (α, t, T)-timed encryption and* MAC *is an existentially unforgeable message authentication code, then* tE-DKE *is adaptively deniable secure against any PPT α-*PRAM *adversary.*

Proof. Assume $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ share an identical sid and also both accepts. Using the notion in the protocol description, $\langle P_i, P_j, E_j[k_1|g^x], C_2\rangle = \langle P_i, P_j, C_1, E_i[k_2|C_1|g^y]\rangle$. Then $\Pi_i^{\ell_i}$ is the initiator and $\Pi_j^{\ell_j}$ is the responder. Also $C_1 = E_j[k_1|g^x]$ and $C_2 = E_i[k_2|C_1|g^y]$. Hence, they have an identical view on (g^x, g^y). Since both accept, both compute $sk = g^{xy}$. This concludes the completeness.

We now consider the adaptive deniability. The oracle simulator is simple: it behaves normally except when he needs to use D_i to decrypt a ciphertext C. In this case, he freezes the adversary $\mathcal{A}$ and uses forced decryption to obtain $m = \mathfrak{T}(C)$. After that, he frees $\mathcal{A}$ and continues the normal execution. Due to the completeness of (α, t, T)-timed encryption, the outcome of $\mathfrak{T}$ differs from $D_i(C)$ only negligibly. Note this holds regardless how C is computed (especially it could be computed under the help of $\mathsf{Execute}$ oracle or even it is invalid). Hence, the simulation is statistically close to the real one. Adaptive deniability follows[1].

Now we consider the authentication property. If $\Pi_t^{\ell_t}$ is the test instance, we need to show that there is a unique partnered instance $\Pi_s^{\ell_s}$ for it. We only need to prove the existence since the probability in an instance to repeatedly sample the same x (or y) is negligible. First consider the case $\Pi_t^{\ell_t}$ is a responder instance and after receiving a round one message C_1^* from a claimed party P_s, he sends $C_2^* = E_s[k_2^*|C_1^*|g^{y^*}]$ to P_s. Denote the event that $\Pi_t^{\ell_t}$ receives a valid third round message τ^* within time t from his sending out C_2^* but no partnered instance in P_s exists for $\Pi_t^{\ell_t}$,

[1] Note that the secrecy of (G, E, D) can be proven in the random oracle model (e.g., tE-RO). It is pointed out in [9] that the deniability in the random oracle model can not be trusted. In our deniability proof here, we only use the property of E that it can be forcefully decrypted by an algorithm $\mathfrak{T}$ in time T. Hence, no secrecy (hence random oracle) is used or considered. Consequently our timed encryption tE-RO in Section 3 can be plugged here.

by Imp. We need to show that $\Pr[\mathsf{Imp}]$ is negligible. Denote the real game by $\varGamma$. We modify $\varGamma$ to $\varGamma'$ such that the game terminates after time t from the moment $\varPi_t^{\ell_t}$ sends out C_2^*. $\Pr[\mathsf{Imp}(\varGamma)] = \Pr[\mathsf{Imp}(\varGamma')]$ since if Imp occurs, it must happen within time t since the moment $\varPi_t^{\ell_t}$ sends out C_2^*. We further modify $\varGamma'$ to $\varGamma''$ such that $E_s[k_2^*|C_1^*|g^{y^*}]$ in $\varPi_t^{\ell_t}$ is replaced by $E_s[\mathbf{0}|C_1^*|g^{y^*}]$. To be consistent, whenever P_s receives the same $E_s[\mathbf{0}|C_1^*|g^{x^*}]$, it proceeds normally with assuming the decryption result being $(k_2^*|C_1^*|g^{y^*})$ and so τ is computed using k_2^*. By reducing to the secrecy property of E under CCA2 attack, we show that $\Pr[\mathsf{Imp}(\varGamma')] = \Pr[\mathsf{Imp}(\varGamma'')] + negl(\kappa)$. Otherwise, an α-PRAM attacker $\mathcal{D}$ for E is described as follows. Assume there are at most μ responder instances. Given a challenge public key E, $\mathcal{D}$ takes $u \leftarrow \{1, \cdots, \mu\}$ and $s \leftarrow \{1, \cdots, n\}$. He sets up the system normally with $\mathcal{A}$ against it, except $E_s = E$. He simulates $\varGamma'$ normally, except when the uth responder instance $\varPi_t^{\ell_t}$ is invoked. In this case, if $\mathsf{pid}_t^{\ell_t} \neq P_s$ or if P_s is corrupted, determinate with 0; otherwise, it takes y^*, k_2^* randomly and outputs the challenge pair as $(\mathbf{0}|C_1^*|g^{y^*}, k_2^*|C_1^*|g^{y^*})$. In turn, he receives C_2^* and sends to P_s. Later $\mathcal{D}$ simulates the game normally, except whenever (1) P_s receives C_2^*, it proceeds using $k_2^*|C_1^*|g^{y^*}$ (e.g. session state $\mathsf{stat}_s^{\ell_s}$ also uses k_2^*); (2) P_s receives $C_2 \neq C_2^*$, he asks his decryption oracle to compute $k_2|C_1|g^y$. Finally, when $P_t^{\ell_t}$ receives a valid round three message τ within time t from the moment its sending C_2^* while no partnered instance in P_s exists for $P_t^{\ell_t}$, then $\mathcal{D}$ outputs 1; otherwise 0. The probability that the uth instance $\varPi_t^{\ell_t}$ is the test instance and P_s is its partner party is $\frac{1}{nu}$. When the guess is correct, C_2^* encoding $\mathbf{0}|C_1^*|g^{y^*}$ (resp. $k_2^*|C_1^*|g^{y^*}$) corresponds to the simulation of $\varGamma''$ (resp. $\varGamma'$). Hence, the non-negligible gap of event in Imp between them implies the non-negligible advantage of $\mathcal{D}$. In addition, since $\mathcal{A}$ is in the α-PRAM model, so is $\mathcal{D}$. $\mathcal{D}$ does not add any delay in converting the challenge/answer of $\mathcal{A}$ to his own. Thus, $\mathcal{D}$ is a valid attacker. This contradicts to the secrecy of E. Finally, $\Pr[\mathsf{Imp}(\varGamma'')]$ is negligible as Imp event (non-partnering) implies that C_1^* in C_2^* was never computed by P_s and hence message C_2 will not be accepted by any instance in P_s. Therefore, the usage of k_2^* in the simulation of $\varGamma''$ is only to evaluate $\mathsf{MAC}_{k_2^*}(\cdot)$ (especially recall $P_t^{\ell_t}$ can not be issued a session state reveal) and hence the non-negligible probability of $\mathsf{Imp}(\varGamma'')$ event can be reduced to break MAC, contradiction! This completes the case $\varPi_t^{\ell_t}$ is a responder. When $\varPi_t^{\ell_t}$ is an initiator, the proof is similar that when $\varPi_t^{\ell_t}$ receives σ, there must exist a unique partnered instance in P_s for it. As a summary, we conclude that $\Pr[\mathsf{Imp}(\varGamma)]$ is negligible and hence the authentication property holds.

We now consider the secrecy property. We have showed that the test instance $\varPi_t^{\ell_t}$ must have a unique partnered instance $\varPi_s^{\ell_s}$. In addition, this $\varPi_s^{\ell_s}$ also only have $\varPi_t^{\ell_t}$ as its unique partnered instance in P_t. This is so due to the fact that sampling a repeated x or y in P_t has a negligible probability (we ignore it). Now we argue the adversary success in the test session is $1/2+negl(\kappa)$; otherwise, an attacker $\mathcal{B}$ can break DDH assumption as follows. Given a challenge tuple (α, β, γ), $\mathcal{B}$ sets up $\varGamma$ normally with $\mathcal{A}$ against it. It takes a distinct pair $u < v$ randomly from $\{1, \cdots, \mu\}$. Then it simulates $\varGamma$ normally, except when the uth instance $\varPi_t^{\ell_t}$ or vth instance $\varPi_s^{\ell_s}$ is invoked. In the former case directly define g^x as α and in the latter case define g^y as β. In this simulation, $x = \log_g \alpha$ and $y = \log_g \beta$ are unknown. Whenever $\varPi_t^{\ell_t}$ or $\varPi_s^{\ell_s}$ is *compromised* or if $\varPi_t^{\ell_t}$ and $\varPi_s^{\ell_s}$ are not partnered instances or if neither of them is chosen as the test session, then aborts; otherwise, define γ as the challenge session key for the test session (which is either $\varPi_t^{\ell_t}$ or $\varPi_s^{\ell_s}$). Finally, output whatever $\mathcal{A}$ does. First of all, we have showed that there is a unique pair of partnered instances, one of which is the test session. If this pair of partnered instance are the uth and vth instances, then the abortion event will not occur. Since prior to the abortion event adversary view is real, the correct guess of (u, v) has a probability of $\frac{1}{\mu(\mu-1)}$. Conditional on the correct guess of (u, v), the adversary view is according to the real distribution in $\varGamma$. Hence, non-negligible advantage of $\mathcal{A}$ in $\varGamma$ implies a non-negligible advantage of $\mathcal{B}$ in his DDH game, contradiction! ■

Corollary 3. *In Theorem 3, if (E, D) is (t, T)-timed encryption, then* tE-DKE *is adaptively deniable secure against any PPT* PRAM *adversary.*

Proof. For any polynomial α, (E, D) is (α, t, T)-timed encryption. By Theorem 3, tE-DKE is adaptively deniable secure against any PPT α-*PRAM* adversary. The claim follows. ■

References

1. M. Bellare, R. Canetti, and H. Krawczyk, a modular approach to the design and analysis of authentication and key exchange protocols, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pp. 419-428, 1998, Dallas, Texas, USA.
2. M. Bellare, P. Rogaway, Entity Authentication and Key Distribution. *CRYPTO 1993*: 232-249.
3. M. Bellare and P. Rogaway, Random Oracle is Practical: A Paradigm for Designing Efficient Protocols, *ACM CCS'93*, pp. 62-73.
4. Dan Boneh, Matthew K. Franklin: Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.* 32(3): 586-615 (2003).
5. R. Canetti and H. Krawczyk, Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels, *Eurocrypt 2001*: 453-474.
6. K. Choo, C. Boyd and Y. Hitchcock, Errors in Computational Complexity Proofs for Protocols, *Advances in Cryptology-ASIACRYPT'05*, B. Roy (Ed.), LNCS 3788, Springer-Verlag, pp. 624-643, 2005.
7. I. F. Blake and A. C.-F. Chan, Scalable, Server-Passive, User- Anonymous Timed Release Public Key Encryption from Bilinear Pairing, in *ICDS 2005: Proceedings of the 25th International Conference on Distributed Computing Systems*, Jun. 2005, pp. 504-513.
8. Dan Boneh and Moni Naor, Timed Commitments and Applications, *CRYPTO'00.*
9. R. Pass, On the deniability in the common reference string and random oracle model, *CRYPTO'03*, pp. 316-337, 2003.
10. J. Cathalo, B. Libert, and J.-J. Quisquater, Efficient and Non-interactive Timed-Release Encryption, in *ICICS 2005: Proceedings of the 7th International Conference on Information and Communications Security*, Dec. 2005, pp. 291-303.
11. Jung Hee Cheon, Nicholas Hopper, Yongdae Kim and Ivan Osipkov, Timed-Release and Key-Insulated Public Key Encryption, *Financial Cryptography 2006.*
12. Jung Hee Cheon, Nicholas Hopper, Yongdae Kim and Ivan Osipkov, Provably Secure Timed-Release Public Key Encryption, *ACM Trans. Inf. Syst. Secur.* 11, 2, Article 8 (May 2008), 44 pages.
13. Giovanni Di Crescenzo, Rafail Ostrovsky, and S. Rajagopalan, Conditional Oblivious Transfer and Timed-Release Encryption, *EUROCRYPT'99.*
14. M. Di Raimondo, R. Gennaro, New Approaches for Deniable Authentication, *ACM CCS'05.*
15. M. Di Raimondo, R. Gennaro, New Approaches for Deniable Authentication, *J. Cryptology* (2009) 22: 572-615.
16. M. Di Raimondo, R. Gennaro and H. Krawczyk, Deniable Authentication and Key Exchange, *ACM CCS'06.*
17. Y. Dodis, D.-H. Yum, Time Capsule Signatures, *Proc. of Financial Cryptography 2005*, 2005.
18. Y. Dodis, J. Katz, A. Smith and S. Walfish, Composability and On-line Deniability of Authentication, *TCC'09.*
19. C. Dwork, M. Naor and A. Sahai, Concurrent Zero-Knowledge, *STOC'98*,pp. 409-418, 1998.
20. J. A. Garay and M. Jakobsson, Timed Release of Standard Digital Signatures, in *FC 2002: Proceedings of the 6th International Conference on Financial Cryptography*, Mar. 2003, pp. 168-182.
21. O. Goldreich, *Foundations of Cryptography: Applications*, Cambridge University Press, 2004.
22. S. Jiang and R. Safavi-Naini, An Efficient Deniable Key Exchange Protocol, *FC'08*, 2008.
23. H. Krawczyk, SKEME, a versatile secure key exchange mechanism for Internet, *NDSS'96*, pp. 114-127.
24. Y. Lindell, A Simpler Construction of CCA2-secure Public-key Encryption under General Assumptions, *EUROCRYPT'03*, pp. 241-254.
25. W. Mao, Timed-Release Cryptography, in *SAC 2001: Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography*, Aug. 2001, pp. 342-357.
26. W. Mao and K. Paterson, On the Plausible Deniability Feature of Internet Protocols, *Manuscript.*
27. T. May, Timed-release crypto, 1993. Available at http://www.cyphernet.org/cyphernomicon/chapter14/14.5.html
28. Kenneth G. Paterson and Elizabeth A. Quaglia, Time-Specific Encryption, *IACR eprint 2010.*
29. R. Rivest, A. Shamir and D. Wagner, Time-lock puzzles and time-release crypto, *unpublished manuscript*, 1996.

Appendix A. One-time Simulation Sound Non-Interactive Zero Knowledge

One-time simualtion sound non-interactive zero knowledge essentially means that except the usual adaptive non-interactive zero knowledge property, the adversary can not prove a false theorem even if the adversary sees a simulated proof for a false theorem. The formulation below essentially follows from [21, 24].

Definition 4. *A pair of probabilistic polynomial time machines (P, V) with a common random string σ is an* adaptive non-interactive zero-knowledge (Adaptive NIZK) *proof system for an $\mathcal{NP}$-language L with $\mathcal{NP}$-relation R_L if the following holds.*

- `Completeness.` *For any $(x, w) \in R_L$, $\Pr[V(x, \sigma, P(x, w, \sigma)) = 1] = 1 - negl(\kappa)$, where σ is uniformly random over $\{0,1\}^{poly(\kappa)}$ and the probability is over the choices of σ and coins of P.*
- `Adaptive Soundness.` *For $\sigma \leftarrow \{0,1\}^\kappa$, the probability that there exists $x \notin L$ and a proof π such that (σ, x, π) is accepting, is negligible, where the probability is over the choices of σ.*
- Adaptive Zero Knowledge. *For any non uniform PPT adversary $\mathcal{A}$, there exists a simulator $\mathcal{SIM}$ such that (σ, x, π) generated in the following two processes are indistinguishable.*
  - ⋄ *Take $\sigma \leftarrow \{0,1\}^\ell$; $(x, w) \leftarrow \mathcal{A}(\sigma)$ s.t. $(x, w) \in R_L$; $\pi = P(x, w, \sigma)$.*
  - ⋄ *$\mathcal{SIM}$ simulates σ with a trapdoor τ; $(x, w) \leftarrow \mathcal{A}(\sigma)$ s.t. $(x, w) \in R_L$; $\mathcal{SIM}$ computes π from (x, σ, τ).*

Adatpive NIZK is one-time simulation sound *if there exists adaptive NIZK simulator $\mathcal{SIM}$ such that for any PPT adversary $\mathcal{A}$ the following holds negligibly.*

- *$\mathcal{SIM}$ outputs (σ, τ) and gives σ to $\mathcal{A}$; $\mathcal{A}(\sigma)$ computes statement x and gives it to $\mathcal{SIM}$; $\mathcal{SIM}$ generates a proof π for x using σ and τ and gives it to $\mathcal{A}$; $\mathcal{A}$ generates a statement x' and its proof π') using (x, σ, π). $\mathcal{A}$ succeeds if $V(x', \sigma, \pi') = 1, (x', \pi') \neq (x, \pi)$ and $x' \notin L$.*

Appendix B. Some Useful Lemmas.

Lemma 2. *Let (G, E, D) be an IND-CPA secure public key encryption. Let $(e, d) \leftarrow G(1^\kappa)$ be a public/private key pair. Then for any c, $\Pr[E_e(D_d(c)) = c] = negl(\kappa)$, where the probability is over the randomness of $E_e(\cdot)$ and $G(1^\kappa)$.*

Proof. Otherwise, assume the claim is violated by a (valid!) ciphertext α. Recall that IND-CPA security states that for **any** m_0, m_1 of the same length, $\Pr[\mathcal{A}(E(m_b)) = b] = 1/2 + negl(\kappa)$. Especially, consider $m_0 = D_d(\alpha)$ and $m_1 = \mathbf{0}$ of the same length. An adversary $\mathcal{A}$ incorporating (m_0, m_1) can break IND-CPA of (E, D) as follows. When receiving $c_b = E_e(m_b)$, $\mathcal{A}$ computes $c \leftarrow E_e(m_0)$. If $c = c_b$, then he outputs 0; otherwise, output 0/1 randomly. Note $c = c_b$ implies $b = 0$ since the decryption of E is unique. Hence, $\Pr[\mathcal{A}(c_b) = b] = 1/2 + \Pr[c = c_b]/2$. On the other hand, $\Pr[c = c_b] = \Pr[b = 0] \cdot \sum_{u \in E_e(m_0)} \Pr^2[E_e(m_0) = u] \geq \Pr^2[E_e(m_0) = \alpha]/2$, non-negligible. Hence, $\Pr[\mathcal{A}(c_b) = b] - 1/2$ is non-negligible, contradiction! ■

Lemma 3. *Let X and Y be random variables over sets $\mathcal{X}, \mathcal{Y}$ respectively and $f : \mathcal{Y} \to \mathcal{Z}$ is a deterministic function, where $\mathcal{Z}$ is an arbitrary set and $\mathcal{X} \subseteq \mathbb{R}$. If $\Pr[X \leq x | Y = y] \leq \lambda$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, then $\Pr[X \leq x | f(Y) = f(y)] \leq \lambda$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

Proof. $\Pr[X \leq x, f(Y) = f(y)] = \sum_{y' \in f^{-1}(f(y))} \Pr[X \leq x, Y = y'] \leq \sum_{y' \in f^{-1}(f(y))} \lambda \Pr[Y = y'] = \lambda \Pr[f(Y) = f(y)]$. The lemma follows. ■

Lemma 4. *Let* $x_0^* \in \{0,1\}^*$ *and* $x_i^* \leftarrow \{0,1\}^a, i = 1, \cdots, n$. $H : \{0,1\}^* \to \{0,1\}^\ell$ *is a random oracle. Consider a set* $\Omega \subseteq \{x_0^* x_1 s \mid s \in \{0,1\}^*, x_1 \in \{0,1\}^a, x_1 \neq x_1^*\}$ *and* $1 \leq j < j' \leq n$. *Fix* $u_0 = x_0^* x_1^* x_2 \cdots x_{j'}$ *for some* $x_i \in \{0,1\}^a \cup \{\textit{empty string}\}$ *but* $x_2 x_3 \cdots x_j \neq x_2^* x_3^* \cdots x_j^*$. *Then,* $x_{j+1}^* \cdots x_n^*$ *is uniformly distributed over* $\{0,1\}^{a(n-j)}$, *given* $x_1^* x_2^* \cdots x_j^*, \Omega, u_0, \{H(x_0^* x_1^* \cdots x_i^*) \mid i = 1, \cdots, n\}, \{H(u) \mid u \in \Omega\}$ *and* $\{H(u_0)\}$.

Proof. Fix $x_0^* x_1^* \cdots x_j^*$. For any possible values of $x_{j+1}^* \cdots x_n^*$, the conditions in the lemma will be valid and sets Ω, $\{u_0\}$ and $\{x_0^* x_1^*, \cdots, x_0^* x_1^* \cdots x_n^*\}$ are mutually disjoint. By the random oracle definition, the randomness of H values $\{H(u) \mid u \in \Omega\}$, $\{H(x_0^* x_1^* \cdots x_i^*) \mid i = 1, \cdots, n\}$ and $\{H(u_0)\}$ are all jointly independent. Hence, they are jointly independent of the randomness for $x_{j+1}^* \cdots x_n^*$ (more specifically, for any value $x_{j+1}^* \cdots x_n^* \in \{0,1\}^{\ell(n-j)}$, the randomness for these H values are unchanged). ■

Lemma 5. *Consider the following game. Take* $x_i^* \leftarrow \{0,1\}^a, i = 1, \cdots, m$. *An adversary* $\mathcal{A}$ *can adaptively query whether any string* $x_1 x_2 \cdots x_i$ *equals* $x_1^* x_2^* \cdots x_i^*$ *for any* $i = 1, \cdots, m$. *Let* $n < m$. $\mathsf{Rev}_N(m, n)$ *denotes the event that, within* N *queries, a query* $x_1^* x_2^* \cdots x_i^*$ *for some* $i > n$ *occurs prior to any query* $x_1^* x_2^* \cdots x_i^*$ *for* $i \leq n$. *Then,* $\Pr[\mathsf{Rev}_N(m, n)] \leq \frac{N}{2^{a(n+1)}}$.

Proof. First of all, $\Pr[\mathsf{Rev}_N(m, n)] \leq \Pr[\mathsf{Rev}_N(n+1, n)]$ since an adversary $\mathcal{A}$ in the game for case $m = n + 1$ can simulate an environment for the case $m > n + 1$ by taking $x_{n+2}^*, \cdots, x_m^*$ normally. In the simulated environment, any query $x_1 \cdots x_i$ for $i \leq n + 1$ is answered by forwarding to $\mathcal{A}$'s challenger; any wuery $x_1 \cdots x_i$ for $i > n + 1$ is handled by querying his own challenger whether $x_1 \cdots x_{n+1}$ is correct. In case of yes and $x_{n+2} \cdots x_i = x_{n+2}^* \cdots x_i^*$ holds as well, return '='; otherwise, return '≠'. The simulation is perfect. Thus, $\mathsf{Rev}_N(m, n)$ for the case $m > n + 1$ implies $\mathsf{Rev}_N(n + 1, n)$. We concentrate on $\Pr[\mathsf{Rev}_N(n + 1, n)]$. $\mathsf{Rev}_N(n + 1, n)$ occurs if and only if query $x_1^* x_2^* \cdots x_{n+1}^*$ occurs before any query $x_1^* x_2^* \cdots x_i^*$ for $i \leq n$. Let p_t be the probability of event that query $x_1^* x_2^* \cdots x_{n+1}^*$ occurs at query t while, before this, no query $x_1^* x_2^* \cdots x_i^*$ occurs for any $i \leq n$. Then $\Pr[\mathsf{Rev}_N(n + 1, n)] = \sum_{t=1}^N p_t$. Now we compute p_t. We consider the candidate set S_ℓ for $x_1^* \cdots x_{n+1}^*$ in view of $\mathcal{A}$ after query ℓ. Initially, no query is issued and $S_0 = \{0,1\}^{a(n+1)}$. At query $\ell < t$, observe that when a query $x_1 x_2 \cdots x_i$ is answered with '≠', then $x_1 x_2 \cdots x_i * ** \neq x_1^* x_2^* \cdots x_{n+1}^*$ and hence this pattern can be removed from $S_{\ell-1}$ and any sequence in S_ℓ satisfies queries issued so far by $\mathcal{A}$ and is a candidate for $x_1^* \cdots x_{n+1}^*$. So $p_t = \frac{|S_1|}{|S_0|} \cdot \frac{|S_2|}{|S_1|} \cdots \frac{|S_{t-1}|}{|S_{t-2}|} \cdot \frac{1}{|S_{t-1}|} = \frac{1}{|S_0|} = 2^{-a(n+1)}$. The lemma follows. ■

Appendix C. Proof of Theorem 2

Proof. *Completeness.* Any normally generated ciphertext in E^* can be correctly decrypted from both the completeness of E and $\mathcal{P}$. For any string $\gamma = (C, \tau, \pi)$, if π is invalid, both $\mathfrak{T}$ and D^* rejects γ. If $(C, \tau) \notin R_e$ (i.e., no (m, r, r') exists for the consistency of (C, τ)), then by soundness of $\mathcal{P}$, π is invalid. Hence, both $\mathfrak{T}$ and D^* rejects. If $(C, \tau) \in R_e$ and π are valid, then $\mathfrak{T}$ uses the forceful open algorithm for `TCom` to obtain m which is identical to $D_d(C)$. Ignoring the time to verify π, $\mathfrak{T}$ runs in time T. T he completeness follows.

Secrecy. Denote the probability of adversary $\mathcal{A}$ outputting 0 in the CCA2 game of (E^*, D^*) by p_{bb} when the challenge bit is b. We also consider the same event in a modified CCA2 game, where the common random string σ is simulated together with a trapdoor η, by p_{bb}^*. Use $p_{bb'}^*$ to denote the same event in the further modified CCA2 game where in the challenge ciphertext (C_b^*, τ^*, π^*), C_b^* encrypts m_b while τ^* commits to $m_{b'}$. (remark: in this case, π^* is simulated using the trapdoor η as normally such a proof does not exist by soundness of $\mathcal{P}$.) The secrecy requires $|p_{00} - p_{11}| = negl(\kappa)$. It suffices to show that $|p_{00} - p_{00}^*|, |p_{00}^* - p_{10}^*|, |p_{10}^* - p_{11}^*|, |p_{11}^* - p_{11}|$ are all negligible. $|p_{bb} - p_{bb}^*|$ is negligible simply due to the adaptive zero knowledge of $\mathcal{P}$. We

now prove the other two. Denote the CCA2 game corresponding to $p_{bb'}$ by $\Gamma_{bb'}$ and to $p^*_{bb'}$ by $\Gamma^*_{bb'}$.
$\underline{|p^*_{00} - p^*_{10}| = negl(\kappa).}$ We show that if this is not true, then (E, D) is not CCA2 secure. Let $\mathcal{A}^*$ be an attacker for (E^*, D^*). The attacker $\mathcal{A}$ for (E, D) can be constructed as follows. Given e, $\mathcal{A}$ simulates σ with trapdoor η and give (e, σ) to $\mathcal{A}^*$. Upon a challenge query (m_0, m_1), he forwards to his own challenger. In turn, it receives C_b. Then he computes $\tau = \texttt{TCom}(m_0)$ and π^* for (C_b, τ^*) using η. Finally, give $\gamma_b = (C_b, \tau^*, \pi^*)$ to $\mathcal{A}^*$. A decryption query (C, τ, π) after the challenge query is handled as follows. By query restriction, $(C, \tau, \pi) \neq (C_b, \tau^*, \pi^*)$. If $(C, \tau, \pi) \neq (C_b, \tau^*, \pi^*)$ but π is invalid, reject normally. If $(C, \tau, \pi) \neq (C_b, \tau^*, \pi^*)$ but π is valid, by one-time simulation soundness of $\mathcal{P}$, it there must exist (m, r, r') such that $((C, \tau), (m, r, r')) \in R_e$. In this case, if $C \neq C_b$, then C is decrypted using the decryption oracle of $\mathcal{A}$ normally; otherwise, we freeze $\mathcal{A}^*$ and then use the forced-open algorithm in $\texttt{TCom}$ to compute the de-commitment m in τ (which is m_b by consistency of C and τ, the validity of π and the one-time simulation soundness of $\mathcal{P}$) and hence b is obtained. Finally, $\mathcal{A}$ outputs whatever $\mathcal{A}^*$ does (the valid output of $\mathcal{A}^*$ must be within time t from receiving his challenge). Note ignoring the negligible decryption error, the view of $\mathcal{A}^*$ is identical to Γ^*_{b0}. Hence, non-negligible gap between p^*_{00} and p^*_{10} implies the non-negligible advantage of $\mathcal{A}$, contradiction to the security of $(E.D)$.
$\underline{|p^*_{10} - p^*_{11}| = negl(\kappa).}$ If this is not true, we show that $\texttt{TCom}$ is not secrecy against a α-PRAM adversary $\mathcal{B}$. $\mathcal{B}$ does the following. He takes (e, d) normally, generates σ with trapdoor η and simulates Γ^*_{10}: upon challenge query (m_0, m_1), he provides (m_0, m_1) to his own challenger and receives $\tau_b = \texttt{TCom}(m_b)$. Then, the challenge for $\mathcal{A}$ is $(E_e(m_0), \tau_b, \pi^*)$, where π^* is simulated using η. Upon a decryption query (c, τ, π), he answers it normally using d. Finally, when $\mathcal{A}$ outputs 0 within time t, $\mathcal{B}$ outputs 0; 1 otherwise. Note the adversary view in the simulation is according to Γ^*_{1b}. In addition, if $\mathcal{A}$ is in the α-PRAM model and outputs b in time t, then $\mathcal{B}$ is in a α-PRAM model and returns b in time t from receiving τ_b (here we ignore the small time to compute $E_e(m_0)$ and π^*). ■