SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2017, Vol. 60 052101:1–052101:19
doi: 10.1007/s11432-015-1037-2

A new construction on randomized message-locked encryption in the standard model via UCEs

Huige WANG[1,2], Kefei CHEN[3]*, Baodong QIN[4], Xuejia LAI[1] & Yunhua WEN[1]

[1]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[2]*Department of Network Engineering, Anhui Science and Technology University, Fengyang* 233100*, China;*
[3]*Department of Mathematics, Hangzhou Normal University, Hangzhou* 311121*, China;*
[4]*Department of Computer Science and Engineering, Southwest University of Science and Technology, Mianyang* 621010*, China*

Received March 4, 2016; accepted April 22, 2016; published online November 14, 2016

Abstract We present a new primitive of randomized message-locked encryption (MLE) in this paper and define a new security model for it. The new primitive, named message-locked encryption3 (hereafter referred as MLE3), is actually a variant of randomized message-locked encryption (Bellare et al. Eurocrypt'13). In order to prevent trivial attacks, our primitive admits a semi-trusted server, which is allowed to hold a secret key of public key encryption (PKE), to verify the correctness of a tag. The new security notion, called privacy chosen-distribution attacks3 (PRV-CDA3), requires that a ciphertext generated by encrypting an unpredictable message and another ciphertext (possible invalid) chosen randomly from a ciphertext space are indistinguishable. Compared with the priori proposed security notion, privacy chosen-distribution attacks (PRV-CDA) (Bellare et al. Eurocrypt'13), which requires that two ciphertexts generated by encrypting two unpredictable messages are indistinguishable, the security notion we propose is much stronger. Based on the new primitive, under the blackbox reductions, we put forward a novel construction which achieves both privacy chosen-distribution attacks3 (PRV-CDA3) and strong tag consistency (STC) securities in the standard model via universal computational extractors (UCEs) (Bellare et al. Crypto'13). In addition, our scheme also provides the validity-testing for ciphertext.

Keywords message-locked encryption3 (MLE3), universal computational extractors (UCEs), privacy chosen-distribution attacks3 (PRV-CDA3), strong tag consistency (STC), standard model

Citation Wang H G, Chen K F, Qin B D, et al. A new construction on randomized message-locked encryption in the standard model via UCEs. Sci China Inf Sci, 2017, 60(5): 052101, doi: 10.1007/s11432-015-1037-2

1 Introduction

Message-locked encryption (MLE), proposed by Bellare et al. [1], is motivated by those file-level deduplication across all their users on cloud computation. When different users upload multiple ciphertexts to the cloud [2,3], message-locked encryption ensures that the server can precisely identify which ciphertexts correspond to encryptions of the same message, so that the server only stores one copy of them. In order to implement this functionality, in message-locked encryption, a tag together with a ciphertext are uploaded to the cloud so that the server can employ the tag to decide whether the ciphertext will

* Corresponding author (email: kfchen@sjtu.edu.cn)

be stored. Moreover, to guarantee that users can decrypt the ciphertexts (possibly different from the original one) corresponding to the same plaintext, message-locked encryption is designed to be a symmetric encryption, where the key is deterministically extracted from the message, aiming to provide a common key for those users. In [4], Douceur et al. proposed a technique called convergent encryption (CE) to capture the functionalities of message-locked encryption. Specifically, when a user Alice wants to upload an encryption of her message M, she first derives a key $K = H(M)$ from her message M and then encrypts M as $c = E(H(M), M)$, where H is a hash function acted as a random oracle and E is a symmetric encryption. Finally the ciphertext c is delivered to the server and the key K is maintained by the user. If the encryption is deterministic, when Bob encrypts the same message M, he would get the same key and the same ciphertext as Alice. Once the server receives c, it can perform deduplication on c as follows: It first checks whether the ciphertext c has been stored or not, if it is, the server will not store it any more, but update database to indicate Bob as an additional user for sharing c. These ideas pave a way for capturing message-locked encryptions. Instances of CE and its variants are widespread in a variety of applications for the purpose of secure deduplication [5–15]. These applications lead to various message-locked encryption schemes, including CE, HCE1, HCE2, RCE, XtCIH, XtDPKE, XtESPKE, SXE schemes in [1], fully randomized MLE2 scheme and deterministic MLE2 scheme in [16] and interactive message-locked encryption scheme in [17].

Overview of PRV-CDA, PRV\$-CDA, TC and STC [1]. In [1], Bellare et al. proposed several underlying security goals for message-locked encryption, including privacy chosen-distribution attacks (PRV-CDA), strong privacy chosen-distribution attacks (PRV\$-CDA), tag consistency (TC) and strong tag consistency (STC). The notion of privacy chosen-distribution attacks (PRV-CDA) is characterized by indistinguishability (IND) between two ciphertexts generated by encrypting two unpredictable messages. While the notion of PRV\$-CDA implements indistinguishability between a random string and a ciphertext generated by encrypting an unpredictable message. Obviously, PRV\$-CDA is stronger than PRV-CDA and both reflect the security in data privacy. TC aims to provide security against attacks in which a legitimate ciphertext is substituted by a forged one but their corresponding tags are equal. In other words, TC guarantees that the message decrypted from the downloaded ciphertext and the one used for generating the corresponding tag are identical. However, TC excludes erasure attacks in which the replaced message is $\perp$. STC also provides security against such erasure attacks besides what TC provides, meaning that an adversary cannot erase a client's message. Clearly, both TC and STC provide data integrity by the consistency of tags. To date, almost all MLEs meet PRV\$-CDA security except schemes XtDPKE, XtESPKE [1] and MLE2 [16]. Again, in tag consistency, except schemes HCE1, HCE2 and RCE [1], all MLEs including MLE2 schemes [16] achieve STC security. From above, we can see that it is easy to construct MLEs with PRV\$-CDA or STC security. However, constructing MLEs that simultaneously meet both securities is not easy. To our knowledge, up to now, only schemes CE, XtCIH and SXE [1] capture the PRV\$-CDA and STC security notions at the same time but all of them are deterministic.

Motivations. In practical contribution, Bellare et al. [1] made ROM security analyses for a class of MLE schemes, in particular, they pointed out that although CE scheme provided both PRV\$-CDA and STC securities, it was less efficient. In addition, as any instantiated schemes in the random oracle model (ROM) have only heuristic security [18], the CE scheme may be insecure in the real world. On the theoretical aspect, Bellare et al. proposed two deterministic MLEs, called as XtCIH and SXE, both of which captured PRV\$-CDA and STC securities in the standard model. However, only SXE captures blackbox reductions and it merely applies for certain class of p-unpredictable message distributions.

From above we can see that it may be difficult to construct a randomized MLEs which not only achieve PRV\$-CDA and STC securities in the standard model via blackbox reductions but have no special restrictions for message distributions. In [1], although Bellare et al. proposed a randomized MLE scheme which eliminated special requirements for message distributions, it can only achieve PRV-CDA and STC securities and we are unaware of whether it can be implemented via blackbox reductions. Then we ask whether there exists randomized MLEs which can capture both PRV\$-CDA and STC securities in the standard model via blackbox reductions and has no restrictions on message distributions. However, to solve this problem is not trivial. The main obstacle is that finding a balance between simultaneously implementing STC and PRV\$-CDA securities for randomized MLEs is hard. To implement STC security, a better solution is designing a valid public algorithm to provide the identical-testing for the plaintext encrypted in ciphertext and that used for generating tag, yet, contradicts with PRV\$-CDA security. Since in the PRV\$-CDA definition, the adversary can trivially break the PRV\$-CDA security by utilizing the identity-testing algorithm to detect whether a given ciphertext is valid. Hence, a natural question arises: can we find an efficient method which may be used to obtain both STC and PRV\$-CDA securities for randomized MLEs in the standard model via blackbox reductions?

In this paper, we present a new primitive, which we call MLE3, a variant of randomized MLE. As a theoretical contribution, in the standard model, we give a construction by using the technique of UCEs suggested by Bellare et al. [19] and prove that it achieves STC and PRV-CDA3 securities via blackbox reductions. Roughly speaking, the notion of the PRV-CDA3 requires that a ciphertext computed by encryption algorithm and the one (possibly invalid) chosen randomly from the ciphertext space are computationally indistinguishable. Clearly, PRV-CDA3 is stronger than PRV-CDA. Compared with existing randomized message-locked encryptions, particularly with Bellare et al.'s XtESPKE [1] scheme, our scheme achieves stronger security in data privacy via blackbox reductions.

Overview of our ideas. In this paper, we first formalize a new prototype, MLE3, then define a new security model, PRV-CDA3, for it. Finally we present a construction for this primitive in the standard model and prove its PRV-CDA3 and STC securities via blackbox reductions. Compared with standard randomized MLE [1], the new primitive differs in two ways. (1) A pair of public/secret key (pk, sk) of public key encryption (PKE) is additionally generated in the parameter generation algorithm, where pk is put in the public parameter and sk is preserved secretly by the server. (2) In order to check the equality of tags, an equality-testing algorithm is designed for this purpose. With such designing, we can guarantee as follows: (1) On the client-side, when the users upload a ciphertext containing a proof of the ciphertext to the server, they can encrypt the proof with this pk and thus prevent the adversary from using this proof to check the validity of this ciphertext; on the server-side, the server can recover the proof with sk corresponding to pk and thus can use this proof to check the correctness of the tag extracted from this ciphertext. (2) Since the randomized algorithm fails to make a direct check about the equality of tags, an equality-testing algorithm for this purpose must be provided. To instantiate MLE3, our construction employs four building blocks: UCE$[\mathsf{S}^{\mathrm{sup}} \cap \mathsf{S}^{q\text{-query}}]$ secure hash function [20], adaptive non-interactive zero-knowledge proof system (NIZK), real or random (ROR) secure symmetric encryption and indistinguishable chosen-plaintext attacks (IND-CPA) secure public key encryption (PKE). In the following, we give a short introduction about the functionalities of the four blocks.

The symmetric encryption (SE) functions in two ways. (1) It is employed to implement the validity-testing of a ciphertext. To do this, we first use a UCE-secure hash function to act on message m and some intermediate results to compute a value t, and then encrypt t and m using the symmetric encryption $\mathcal{SE}$ to obtain c_2. In turn, when executing the validity-testing for a ciphertext with the form (c_1, c_2), we first decrypt c_2 to recover t' and m' and then recompute t using the recovered message m' and the intermediate values computed in the decryption process. Finally we check the validity of the ciphertext by testing whether $t = t'$. (2) The ROR security of the symmetric encryption helps implementing the PRV-CDA3 security of the scheme. The reason is that the ciphertext c_2, determined by the symmetric encryption, is indistinguishable from a random string.

The NIZK is used for verifying the correctness of a tag. Roughly speaking, for a statement such as $x = (\tau, c_1, c_2) \in \mathcal{L}$ with witness $w = (m, r, L)$, we first compute a proof $\pi = \mathcal{P}(x, w)$ using the prover algorithm of the NIZK, where r and L are random numbers used for generating τ and c_1, and m is the message encrypted in c_2. Then, the proof π is encrypted with the public key pk of PKE. When the server receives ciphertext $c = (\tau, c_1, c_2, v)$, it first decrypts v to recover π under the secret key sk corresponding to pk and then checks the correctness of the tag τ by verifying whether $\mathcal{V}(x, \pi) = 1$. Clearly, to achieve strong tag consistency (STC), the values r, L and m used for generating τ, c_1 and c_2 must be identical. In other words, for any $x \notin \mathcal{L}$, there exists no efficient algorithm that can forge a π to make $\mathcal{V}(x, \pi) = 1$. Actually, the soundness of the NIZK just guarantees this. In addition, we stress that encrypting π is necessary, because once π is leaked to adversary, it can use the verification algorithm $\mathcal{V}(x,\pi)$ to check the validity of the ciphertext and thus breaks the PRV-CDA3 security.

Recently, Brzuska et al. [21] pointed out that if indistinguishability obfuscation (IO) exists, computationally unpredictable UCE[$\mathsf{S}^{\mathrm{cup}}$] secure hash function is impossible, while statistically unpredictable UCE[$\mathsf{S}^{\mathrm{sup}}$] secure hash function is not the case. In order to analyze the unpredictability of the source S in the sense of information theoretic, our construction applies a statistically unpredictable UCE[$\mathsf{S}^{\mathrm{sup}} \cap \mathsf{S}^{q\text{-query}}$] secure hash function, which only allows making at most q queries [20], to instantiate our scheme. In fact, since our scheme does not rely on IO assumption, any UCE secure hash function may be applied to our scheme as long as its construction exists. In the security proofs, we show that the statistical unpredictability of the source S follows from the guessing probability $\mathsf{Guess}_{\mathcal{M}}(\cdot)$ and the negligible value $\frac{2}{2^{l(\lambda)}} + \frac{\mathsf{Guess}_{\mathcal{M}}(\lambda)}{2^{2l(\lambda)}}$.

Note that although the new construction obtains more advantages than existing schemes in security, we are still unable to provide PRV\$-CDA security. First, review that the PRV\$-CDA requires that the encryption of a message is indistinguishable from a random string. However, the tag τ in a ciphertext fails to satisfy this requirement, because the well-formed structure $(g^r, g^{r\mathsf{H.Eval}(1^\lambda,hk,K,1^{l_p(\lambda)})})$ of the tag, as an element in the group tuple (G,G), is hard to transform into a random string. In addition, to find a PKE with ciphertexts indistinguishable from a random string is also difficult.

2 Preliminaries

Notations and conventions. Throughout this paper, "PPT" stands for "probabilistic polynomial-time", $\mathbb{N}$ denotes the natural number set and $\lambda \in \mathbb{N}$ denotes the security parameter. $y \leftarrow A(x_1,\ldots;R)$ denotes that the algorithm A on inputting $x_1,\ldots$ and coins R, outputs y. For simplicity, $y \leftarrow A(x_1,\ldots;R)$ is written as $y \xleftarrow{\$} A(x_1,\ldots)$ with implied coins. If $n \in \mathbb{N}$, then it denotes the set $\{1,\ldots,n\}$. A function negl is negligible in λ if $\mathrm{negl}(\lambda) \in \lambda^{-\omega(1)}$ and a function Poly a polynomial if $\mathrm{Poly} \in \lambda^{\mathcal{O}(1)}$. If X denotes a random variable over a set S, then $\max_a \Pr[X=a]$ denotes the predictability of X and $-\log(\max_a \Pr[X=a])$ denotes the min-entropy $\mathsf{H}_\infty(X)$ of X. If $\boldsymbol{M}$ is a vector, then $|\boldsymbol{M}|$ is the number of elements in $\boldsymbol{M}$. In this paper, we use the game framework from [22,23]. If G denotes a game equipped with a main procedure and some other procedures, then the game G starts by running the main procedure in which an adversary A is run and allowed to access some oracles after some initializations. Finally, when A finishes its execution, the game G continues to execute over $A's$ outputs and returns the final results. Let $\mathsf{G}^A \Rightarrow y$ denote that G is executed with A to output y. Generally, $\mathsf{G}^A \Rightarrow \mathrm{true}$ is abbreviated as G.

2.1 Universal computational extractors (UCEs)

In this section, we give the definition of UCE from [19].

Definition 1 (UCE). Let $\mathsf{H} = (\mathsf{H.KGen};\mathsf{H.Eval};\mathsf{H.kl};\mathsf{H.il};\mathsf{H.ol})$ be a hash function family and let (S,D) be two-stage adversaries, where $\mathsf{H.KGen}$ denotes the key generation algorithm which on inputting 1^λ, outputs a key $hk \in \{0,1\}^{\mathsf{H.kl}}$, $\mathsf{H.Eval}$ denotes the evaluation algorithm which on inputting 1^λ, hk, $x \in \{0,1\}^{\mathsf{H.il}}$ and $1^{\mathsf{H.ol}}$, outputs $y \in \{0,1\}^{\mathsf{H.ol}}$. The advantage with respect to (S,D) in game UCE is defined as

$$\mathsf{Adv}^{\mathsf{UCE}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda) := 2.\Pr[\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)] - 1.$$

Without any restrictions, (S,D) can easily get a correct guess. In order to prevent such an attack, Bellare et al. [19] impose some restrictions on S which yield a variety of UCE classes, including UCE[$\mathsf{S}^{\mathrm{cup}}$] and UCE[$\mathsf{S}^{\mathrm{sup}}$].

Unpredictability. A source S is defined as computationally unpredictable source $\mathsf{S}^{\mathrm{cup}}$ if the advantage with respect to any PPT predictor P in game PRED,

$$\mathsf{Adv}^{\mathsf{PRED}}_{\mathsf{S},\mathsf{P}}(\lambda) := \Pr[\mathrm{PRED}^{\mathsf{P}}_{\mathsf{S}}(\lambda)] - 1$$

is negligible. If the predictor P is allowed to run in unbounded time, then S is defined as the statistically unpredictable source $\mathsf{S}^{\rm sup}$.

```
Game $\mathsf{UCE}_{\mathsf{H}}^{\mathsf{S},\mathsf{D}}(\lambda)$
$b \xleftarrow{\$} \{0,1\}$;
$hk \xleftarrow{\$} \mathsf{H.KGen}(1^\lambda)$;
$L \xleftarrow{\$} \mathsf{S}^{\mathsf{HASH}}(1^\lambda)$;
$b' \xleftarrow{\$} \mathsf{D}(1^\lambda, hk, L)$;
Return $b = b'$.

$\mathsf{HASH}(x, 1^l)$
If $T[x,l] = \perp$ then
    If $b = 1$ then $T[x,l] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, x, 1^l)$;
    else $T[x,l] \xleftarrow{\$} \{0,1\}^l$;
Return $T[x,l]$.
```

```
Game $\mathsf{PRED}_{\mathsf{S}}^{\mathsf{P}}(\lambda)$
finish $\leftarrow$ false, $\mathcal{X} \leftarrow \emptyset$;
$L \xleftarrow{\$} \mathsf{S}^{\mathsf{HASH}}(1^\lambda)$;
finish $\leftarrow$ true;
$\mathcal{X}' \xleftarrow{\$} \mathsf{P}^{\mathsf{HASH}}(1^\lambda, L)$;
Return $(\mathcal{X} \cap \mathcal{X}' \neq \emptyset)$.

$\mathsf{HASH}(x, 1^l)$
If finish = false, then $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$;
If $T[x,l] = \perp$ then $T[x,l] \xleftarrow{\$} \{0,1\}^l$;
Return $T[x,l]$.
```

The source S is defined as strong statistically unpredictable source $\mathsf{S}^{s\text{-sup}}$ if the predictor is additionally provided the query results in the following game.

```
Game $\mathsf{stPRED}_{\mathsf{S}}^{\mathsf{P}}(\lambda)$
$\mathcal{X}, \mathcal{Y} \leftarrow \emptyset$;
finish $\leftarrow$ false;
$L \xleftarrow{\$} \mathsf{S}^{\mathsf{HASH}}(1^\lambda)$;
finish $\leftarrow$ true;
$x' \xleftarrow{\$} \mathsf{P}^{\mathsf{HASH}}(1^\lambda, L, \mathcal{Y})$;
Return $(x' \in \mathcal{X})$.
```

```
$\mathsf{HASH}(x, 1^l)$
If finish = false, then $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$;
$y \xleftarrow{\$} \{0,1\}^{\mathsf{H.ol}(\lambda)}$;
$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y\}$;
Return $y$.
```

A construction for UCE secure hash function [20]. Here, we review a construction with strong statistical unpredictability and q-query source in [20] which is proved to be $\mathsf{UCE}[\mathsf{S}^{s\text{-sup}} \cap \mathsf{S}^{q\text{-query}}]$ secure under the assumptions of secure puncturable pseudorandom function (PRF) [24], secure IO [25] and secure q-composable VGB point obfuscator [26]. Then construction is defined as below:

```
$\mathsf{H.KGen}(1^\lambda)$
$K_{\mathsf{F}} \xleftarrow{\$} \mathsf{F.Kg}(1^\lambda)$;
$hk \xleftarrow{\$} \mathsf{IO}(\mathsf{Pad}(s(\lambda), \mathsf{F.Eval}(K_{\mathsf{F}}, \cdot)))$;
Return $hk$.
```

```
$\mathsf{H.Eval}(hk, x)$
$\bar{\mathsf{P}} \leftarrow hk$;
Return $\bar{\mathsf{P}}(x)$.
```

In the construction $\mathsf{Pad} : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ and F is a puncturable PRF defined as in [24]. If the readers want to learn its detailed definition, they can refer to [24].

Remark 1. Obviously, a $\mathsf{UCE}[\mathsf{S}^{\rm sup} \cap \mathsf{S}^{q\text{-query}}]$ secure hash function can be viewed as a slightly weaker version of a $\mathsf{UCE}[\mathsf{S}^{s\text{-sup}} \cap \mathsf{S}^{q\text{-query}}]$ secure hash function [20]. In particular, both classes of functions have fixed input length and fixed output length. However, the technique of [19] can be used to convert $\mathsf{UCE}[\mathsf{S}^{\rm sup} \cap \mathsf{S}^{q\text{-query}}]$ secure hash functions with fixed input length and fixed output length into ones with fixed input length and variable output length. In our construction, a $\mathsf{UCE}[\mathsf{S}^{\rm sup} \cap \mathsf{S}^{q\text{-query}}]$ secure hash function with fixed input length and variable output length is applied.

2.2 Adaptive non-interactive zero-knowledge proof system (NIZK)

In the following, we give a definition of adaptive NIZK.

```
$\underline{\mathsf{ROR}^{A}_{\mathcal{SE}}(1^\lambda)}$
$b \xleftarrow{\$} \{0,1\}$, $b' \xleftarrow{\$} A^{\mathsf{Enc1}(\cdot)}(1^\lambda)$. Return $b = b'$.
$\underline{\mathsf{Enc1}(M)}$
$\kappa \xleftarrow{\$} \mathcal{SE}.\mathcal{G}(1^\lambda)$. If $b = 1$ then $c \xleftarrow{\$} \mathcal{SE}.\mathcal{E}_\kappa(M)$ else $c \xleftarrow{\$} \{0,1\}^{|c|}$. Return $c$.
```

Figure 1 ROR game.

```
$\underline{\mathsf{IND\text{-}CPA}^{A}_{\mathcal{PE}}(1^\lambda)}$
$(pk, sk) \xleftarrow{\$} \mathcal{KG}(1^\lambda)$, $b \xleftarrow{\$} \{0,1\}$, $b' \xleftarrow{\$} A^{\mathsf{Enc2}(\cdot,\cdot)}(1^\lambda, pk)$. Return $b = b'$.
$\underline{\mathsf{Enc2}(M_0, M_1)}$
If $b = 1$ then $c \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, M_1)$ else $c \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, M_0)$. Return $c$.
```

Figure 2 IND-CPA game.

Definition 2 (Adaptive NIZK). This pair of algorithms $(\mathcal{P}, \mathcal{V})$ is an adaptive NIZK for a language $\mathcal{L} \in \mathsf{NP}$ if it satisfies the properties of completeness, soundness and zero-knowledge. The completeness says that for all common random string CRS $\leftarrow \{0,1\}^{\mathsf{Poly}(\lambda)}$, all statement $x \in \mathcal{L}$ with a witness w, the algorithm $\mathcal{P}(\mathrm{CRS}, x, w)$ generates a proof π which makes $\mathcal{V}(\mathrm{CRS}, x, \pi) = 1$ hold. The soundness says that after seeing CRS, no cheating prover $\mathcal{P}^*$ can generate a pair(x, π) with $x \notin \mathcal{L}$ which makes the algorithm $\mathcal{V}(\mathrm{CRS}, x, \pi) = 1$ hold with non-negligible probability. The advantage that $\mathcal{P}^*$ breaks the soundness associated with $(\mathcal{P}, \mathcal{V})$ can be written as $\mathsf{Adv}^{\mathsf{sound}}_{(\mathcal{P},\mathcal{V}),\mathcal{P}^*} \leqslant \mathrm{negl}(\lambda)$. The zero-knowledge says that given a simulated CRS, which is generated by a simulator $\mathsf{Sim}_1(1^\lambda)$, and a simulated proof π, which is generated by a simulator $\mathsf{Sim}_2(x)$, the simulators $(\mathsf{S}_1, \mathsf{S}_2)$ can simulate the view of adversaries (A_1, A_2). A_1 is the algorithm that on inputting CRS $\leftarrow \{0,1\}^{\mathsf{Poly}(\lambda)}$, outputs (x, w); A_2 is the algorithm that on inputting CRS and (x, w), outputs a proof π. The advantage that (A_1, A_2) can break the zero-knowledge with respect to $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ can be defined as $\mathsf{Adv}^{\mathsf{ZK}}_{(\mathcal{P},\mathcal{V}),(A_1,A_2)} \leqslant \mathrm{negl}(\lambda)$.

2.3 Symmetric encryption

A symmetric encryption (SE) is a tuple of three algorithms $(\mathcal{SE}.\mathcal{G}, \mathcal{SE}.\mathcal{E}, \mathcal{SE}.\mathcal{D})$. The algorithm $\mathcal{SE}.\mathcal{G}$ on inputting 1^λ, outputs the key κ. The algorithm $\mathcal{SE}.\mathcal{E}$ on inputting a key κ and a plaintext $M \in \mathcal{M}$, outputs a ciphertext $c = \mathcal{SE}.\mathcal{E}(k, M)$, where $\mathcal{M}$ denotes the plaintext space. The algorithm $\mathcal{SE}.\mathcal{D}$ on inputting the key κ and c, returns a plaintext $M \in \mathcal{M}$ or $\perp$. For correctness, the equation $\mathcal{SE}.\mathcal{D}_\kappa(\mathcal{SE}.\mathcal{E}_\kappa(M)) = M$ holds for all $\kappa \xleftarrow{\$} \mathcal{SE}.\mathcal{G}(1^\lambda)$ and all $M \in \mathcal{M}$. Meanwhile, we require that $\mathcal{SE}$ should achieve the real or random (ROR) security defined below.

Definition 3 (ROR security of $\mathcal{SE}$). A symmetric encryption $\mathcal{SE}$ is ROR secure if for all PPT adversary A, the advantage $\mathsf{Adv}^{\mathsf{ROR}}_{\mathcal{SE},A}(\lambda) = 2.\Pr[\mathsf{ROR}^{A}_{\mathcal{SE}}(1^\lambda)] - 1 \leqslant \mathrm{negl}(\lambda)$ holds. Below define the game $\mathsf{ROR}^{A}_{\mathcal{SE}}(1^\lambda)$ in Figure 1.

2.4 Public key encryption (PKE)

A PKE consists of three algorithms $(\mathcal{PE}.\mathcal{G}, \mathcal{PE}.\mathcal{E}, \mathcal{PE}.\mathcal{D})$. The algorithm $\mathcal{PE}.\mathcal{G}$ on inputting 1^λ, outputs a pair (pk, sk) which respectively denotes the public key and private key. The algorithm $\mathcal{PE}.\mathcal{E}$ on inputting pk and a plaintext $M \in \mathcal{M}$, outputs a ciphertext c, where $\mathcal{M}$ denotes the message space. The algorithm $\mathcal{PE}.\mathcal{D}$ on inputting sk and c, outputs a plaintext $M \in \mathcal{M}$ or $\perp$. The correctness requires that $\mathcal{PE}.\mathcal{D}(sk, \mathcal{PE}.\mathcal{E}(pk, M)) = M$ is satisfied for all $(pk, sk) \xleftarrow{\$} \mathcal{PE}.\mathcal{G}(1^\lambda)$ and $M \in \mathcal{M}$. We require that $\mathcal{PE}$ should achieve the indistinguishability chosen-plaintext (IND-CPA) security defined below.

Definition 4 (IND-CPA security of $\mathcal{PE}$). A PKE $\mathcal{PE}$ is IND-CPA secure if for all PPT adversary A, the advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{PE},A}(\lambda) = 2.\Pr[\mathsf{IND\text{-}CPA}^{A}_{\mathcal{PE}}(1^\lambda)] - 1 \leqslant \mathrm{negl}(\lambda)$ holds. Below define the game $\mathsf{IND\text{-}CPA}^{A}_{\mathcal{PE}}(1^\lambda)$ in Figure 2.

3 A new variant of randomized message-locked encryption (MLE3)

3.1 Defining MLE3

Usually, a standard randomized message-locked encryption is equipped with five algorithms: $\mathcal{ME}.\mathcal{PG}$, $\mathcal{ME}.\mathcal{KD}$, $\mathcal{ME}.\mathcal{E}$, $\mathcal{ME}.\mathcal{TG}$ and $\mathcal{ME}.\mathcal{D}$. The parameter generation algorithm $\mathcal{ME}.\mathcal{PG}$ is a PPT algorithm which on inputting 1^λ, outputs a public parameter pp; The key derivation algorithm $\mathcal{ME}.\mathcal{KD}$ is defined as a deterministic algorithm which on inputting pp and a plaintext m, outputs a symmetric encryption key K; The encryption algorithm $\mathcal{ME}.\mathcal{E}$ is defined as a PPT algorithm which on inputting pp, K and m, returns a ciphertext c; The tag generation algorithm $\mathcal{ME}.\mathcal{TG}$ is defined as a deterministic algorithm which on inputting pp and c, returns a tag T; The decryption algorithm $\mathcal{ME}.\mathcal{D}$ is also a deterministic algorithm which on inputting pp, k and c, returns m or $\perp$. The new variant of randomized message-locked encryption, MLE3, is defined almost identical to the standard randomized message-locked encryption except that an equality testing algorithm $\mathcal{ME}.\mathcal{ET}$ is additionally defined and a little change occurs on the algorithms $\mathcal{ME}.\mathcal{PG}$ and $\mathcal{ME}.\mathcal{TG}$. In the following, we only give the descriptions of the changed algorithms $\mathcal{ME}.\mathcal{PG}$ and $\mathcal{ME}.\mathcal{TG}$ and the added $\mathcal{ME}.\mathcal{ET}$.

- $\mathcal{ME}.\mathcal{PG}$: In a standard randomized message-locked encryption, this algorithm $\mathcal{ME}.\mathcal{PG}$ may be executed by a user or fully-trusted server, while in MLE3, a semi-trusted server (here the "semi-trusted" means that the server is trusted with respect to sk (since it is given sk and is not expected to share it), but is not trusted with respect to user's messages (e.g., contents of the files)) is arranged to run this algorithm which on inputting 1^λ, outputs the public parameter pp. Besides, it additionally executes the key generation algorithm $\mathcal{PE}.\mathcal{G}$ of the public-key encryption $\mathcal{PE}$ to generate a pair of public and secret key (pk, sk) with pk involved in pp and sk preserved by the server.
- $\mathcal{ME}.\mathcal{TG}$: This algorithm is executed by the server. Besides taking the system parameter pp and a ciphertext c as inputs, the secret key sk is also involved in the inputs, finally a valid tag T is output.
- $\mathcal{ME}.\mathcal{ET}$: This is an equality testing algorithm for tags, which takes as input the system parameter pp, two valid tags T_1 and T_2 and outputs a boolean value true or false indicating whether the two tags correspond to the same plaintext or not.

Priori to defining the PRV-CDA3 security, we first give some basic notions. The guessing probability $\mathsf{Guess}_{\mathcal{M}}$ over a plaintext distribution $\mathcal{M}$ is defined as $\mathsf{Guess}_{\mathcal{M}} := \max\{\Pr[\boldsymbol{M}[i] = M]\}$, where the probability conditions on all i, M and $\boldsymbol{M} \leftarrow_\$ \mathcal{M}$. Moreover, if the guessing probability $\mathsf{Guess}_{\mathcal{M}}$ is negligible, we say that each component $\boldsymbol{M}[i]$ of the message vector $\boldsymbol{M}$ has min-entropy $-\log \mathsf{Guess}_{\mathcal{M}}$. Here, we require that the message vector should satisfy: (1) $|\boldsymbol{M}| \leqslant v(\lambda)$ and $|\boldsymbol{M}[i]| \in \mathcal{ME}.\mathcal{IL}(\lambda)$ for each $i \in [|\boldsymbol{M}|]$ and all polynomial $v(\lambda)$. (2) All $\boldsymbol{M}[1], \ldots, \boldsymbol{M}[|\boldsymbol{M}|]$ are different. $\mathcal{ME}.\mathcal{IL}(\cdot)$ denotes the input-length function of scheme $\mathcal{ME}$.

In Figure 3, we respectively give the definitions of strong tag consistency (STC) and privacy chosen-distribution attacks3 (PRV-CDA3).

Definition 5 (PRV-CDA3 security of $\mathcal{ME}$)**.** A message-locked encryption $\mathcal{ME}$ is PRV-CDA3 secure if for all PPT adversary A, the advantage $\mathsf{Adv}^{\mathsf{PRV\text{-}CDA3}}_{\mathcal{ME},\mathcal{M},A}$=2.$\Pr[\mathsf{PRV\text{-}CDA3}^{A}_{\mathcal{ME},\mathcal{M}}(\lambda)] - 1 \leqslant \mathrm{negl}(\lambda)$.

In game PRV-CDA3, we assume that the ciphertext space CiphS has the form

$$\mathrm{CiphS} = (\mathrm{CiphS}_1, \ldots, \mathrm{CiphS}_n).$$

Note that for each $j \in [n]$, the component CiphS_j in CiphS is a valid ciphertext space corresponding to that of the sub-encryption algorithm implied in $\mathcal{ME}$. In addition, we assume that there are two participants in this game, namely the adversary A and the challenger $\mathcal{C}$. At the beginning, the challenger $\mathcal{C}$ runs the algorithm $\mathcal{ME}.\mathcal{PG}(1^\lambda)$ to generate a pair of system parameter and secret key (pp, sk) and sends pp to the adversary A and then chooses a random bit b. When $\mathcal{C}$ receives a plaintext distribution $\mathcal{M}$ from the adversary A, $\mathcal{C}$ samples a pair of plaintext vector and auxiliary input $(\boldsymbol{M}, Z)$ from $\mathcal{M}$. Subsequently, $\mathcal{C}$ encrypts $\boldsymbol{M}$ as in Figure 3: For each $i \in [|\boldsymbol{M}|]$, $\mathcal{C}$ computes the key $\boldsymbol{K}[i] \leftarrow \mathcal{ME}.\mathcal{KD}(pp, \boldsymbol{M}[i])$. When $b = 1$, $\mathcal{C}$ computes the ciphertext $\boldsymbol{c}[i]$ by invoking the algorithm $\mathcal{ME}.\mathcal{E}(pp, \boldsymbol{K}[i], \boldsymbol{M}[i])$; otherwise, $\mathcal{C}$ generates $\boldsymbol{c}[i]$ in the way of $\boldsymbol{c}[i] \overset{\$}{\leftarrow} \mathrm{CiphS}$. Finally, $\mathcal{C}$ sends the ciphertext vector $\boldsymbol{c}$ and the auxiliary input

```
Game $\mathsf{PRV\text{-}CDA3}^{A}_{\mathcal{ME},\mathcal{M}}(\lambda)$
$(pp, sk) \xleftarrow{\$} \mathcal{ME}.\mathcal{PG}(1^{\lambda})$;
$b \xleftarrow{\$} \{0,1\}$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^{\lambda})$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
    $\boldsymbol{K}[i] \leftarrow \mathcal{ME}.\mathcal{KD}(pp, \boldsymbol{M}[i])$;
    If $b = 1$ then $\boldsymbol{c}[i] \xleftarrow{\$} \mathcal{ME}.\mathcal{E}(pp, \boldsymbol{K}[i], \boldsymbol{M}[i])$;
    If $b = 0$ then $\boldsymbol{c}[i] \xleftarrow{\$}$ CiphS;
$b' \xleftarrow{\$} A(pp, \boldsymbol{c}, Z)$;
Return $(b = b')$;
```

```
Game $\mathsf{STC}^{A}_{\mathcal{ME}}(\lambda)$
$(pp, sk) \xleftarrow{\$} \mathcal{ME}.\mathcal{PG}(1^{\lambda})$;
$(M, C') \xleftarrow{\$} A(pp)$;
If $M = \perp$ or $C' = \perp$ return false.
$K \leftarrow \mathcal{ME}.\mathcal{KD}(pp, M)$;
$C \xleftarrow{\$} \mathcal{ME}.\mathcal{E}(pp, K, M)$;
$T \leftarrow \mathcal{ME}.\mathcal{TG}(pp, C, sk)$;
$T' \leftarrow \mathcal{ME}.\mathcal{TG}(pp, C', sk)$;
$M' \leftarrow \mathcal{ME}.\mathcal{D}(pp, K, C')$;
If $\mathcal{ME}.\mathcal{ET}(pp, T', T) =$ true and $M \neq M'$ return true
else return false.
```

Figure 3 PRV-CDA3 game and STC game.

Z to the adversary A which outputs b' as the guess of the bit b.

Obviously, when $b = 1$, $\boldsymbol{c}[i] = (\boldsymbol{c}_1[i], \ldots, \boldsymbol{c}_n[i]) = \mathcal{ME}.\mathcal{E}(pp, \boldsymbol{K}[i], \boldsymbol{M}[i])$ is a valid ciphertext for a plaintext vector $\boldsymbol{M}$; when $b = 0$, with overwhelming probability, the challenge ciphertext $\boldsymbol{c}$ is invalid, namely, $\boldsymbol{c}[i] = (\boldsymbol{c}_1[i], \ldots, \boldsymbol{c}_n[i])$ with $\boldsymbol{c}_1[i] \xleftarrow{\$} \mathrm{CiphS}_1, \ldots, \boldsymbol{c}_n[i] \xleftarrow{\$} \mathrm{CiphS}_n$. Therefore, PRV-CDA3 implies indistinguishability between a ciphertext computed by encrypting an unpredictable message and the one chosen randomly from the ciphertext space CiphS. In addition, note that if each component CiphS_j in CiphS may be encoded into a valid bit string, then the PRV-CDA3 notion can be converted into the PRV\$-CDA notion. It might be of independent interest to research this case.

Definition 6 (STC security of $\mathcal{ME}$)**.** A message-locked encryption scheme $\mathcal{ME}$ is STC secure if for all PPT adversary A, the advantage $\mathsf{Adv}^{\mathsf{STC}}_{\mathcal{ME},A} = \Pr[\mathsf{STC}^{A}_{\mathcal{ME}}(\lambda)] \leqslant \mathrm{negl}(\lambda)$.

Similarly, we assume that the game STC is played between an adversary A and a challenger $\mathcal{C}$. At the beginning, the challenger $\mathcal{C}$ first runs the algorithm $\mathcal{ME}.\mathcal{PG}(1^{\lambda})$ to generate a pair of system parameter and secret key (pp, sk) and sends pp to the adversary A. On inputting pp, A outputs a pair of plaintext and a forged ciphertext (M, C'). If $M = \perp$ or $C' = \perp$, the game returns false; otherwise, $\mathcal{C}$ computes a key $K \leftarrow \mathcal{ME}.\mathcal{KD}(pp, M)$, a ciphertext $C \xleftarrow{\$} \mathcal{ME}.\mathcal{E}(pp, K, M)$, a real tag $T \leftarrow \mathcal{ME}.\mathcal{TG}(pp, C, sk)$, a forged tag $T' \leftarrow \mathcal{ME}.\mathcal{TG}(pp, C', sk)$ and a plaintext $M' \leftarrow \mathcal{ME}.\mathcal{D}(pp, K, C')$. Finally, the challenger $\mathcal{C}$ checks whether the real tag T and the forged T' satisfy the equation $\mathcal{ME}.\mathcal{ET}(pp, T', T) =$ true and whether the inequation $M \neq M'$ holds, if so, $\mathcal{C}$ outputs true as the final output of the game. In addition, note that if the condition statement "If $\mathcal{ME}.\mathcal{ET}(pp, T', T) =$ true and $M \neq M'$" is replaced with "If $\mathcal{ME}.\mathcal{ET}(pp, T', T) =$ true and $M \neq M'$ and $M \neq \perp$", then the STC security definition becomes the TC security definition.

Remark 2. Though the new primitive allows a pair of public/secret key (pk, sk) to be generated in the parameter generation algorithm, we stress that this assumption would not degrade the security-level, conversely, it plays a crucial role in guaranteeing the correctness of tags. Because when a ciphertext is uploaded to the server, we let the server to execute all operations on the tags, including extracting tags from ciphertexts with the secret key sk, however, exactly these operations give the server the ability to check the correctness of the tags. In addition, note that sk should not be leaked, as otherwise, the adversary may use it to break the PRV-CDA3 security. Furthermore, we stress that sk cannot be preserved by the user. This is because once the user owns the secret key sk, the tag can only be recovered with sk on the client-side, however, when the tag arrives at the server-side, it may have been tampered during transmission on the Internet, and thus leads to the server receiving an incorrect tag. Finally, note that since the public key encryption $\mathcal{PE}$ is not used elsewhere except encrypting π, so the full scheme is still a symmetric encryption.

3.2 Construction

Below, we give a concrete construction of $\mathcal{ME}$.

Let GroupGen be a PPT algorithm that on inputting 1^{λ}, outputs a group description $\mathcal{G} = (G, G_T, p, g, e)$, where G and G_T are groups with large prime order p, the element $g \in G$ is a generator of G and $e : G \times G \rightarrow G_T$ is an efficiently computable nondegenerate bilinear map. Let p be $l_p(\lambda)$-bit prime, then our randomized message-locked encryption scheme $\mathcal{ME} = (\mathcal{ME.PG}, \mathcal{ME.KD}, \mathcal{ME.E}, \mathcal{ME.D}, \mathcal{ME.TG}, \mathcal{ME.ET})$ with input-length $\mathcal{ME.IL}(\lambda)$ and output length $\mathbb{N}$ comprises the following building blocks.

- An adaptive NIZK $\mathcal{ZK} = (\mathcal{P}, \mathcal{V})$.
- A ROR secure SE scheme $\mathcal{SE} = (\mathcal{SE.G}, \mathcal{SE.E}, \mathcal{SE.D})$.
- A $\mathsf{UCE}[\mathsf{S}^{\text{sup}} \cap \mathsf{S}^{q\text{-query}}]$ secure hash function H with fixed input length $\mathsf{H.il}(\lambda) = 2l_p(\lambda) + 3l(\lambda) + \mathcal{ME.IL}(\lambda)$ and variable output length $\mathsf{H.ol}(\lambda) = \mathbb{N}$.
- An IND-CPA secure PKE scheme $\mathcal{PE} = (\mathcal{PE.G}, \mathcal{PE.E}, \mathcal{PE.D})$.

Notes. If the size of the input to H is less than $2l_p(\lambda) + 3l(\lambda) + \mathcal{ME.IL}(\lambda)$ bits, then it is padded to $2l_p(\lambda) + 3l(\lambda) + \mathcal{ME.IL}(\lambda)$ bits with some zero-bits from the highest significant bit of the input.

Parameter generation. $\mathcal{ME.PG}(1^\lambda)$ is executed by a semi-trusted server. On inputting 1^λ, the algorithm first samples $(G, G_T, p, g, e) \xleftarrow{\$} \text{GroupGen}(1^\lambda)$. Then it chooses a $\mathsf{UCE}[\mathsf{S}^{\text{sup}} \cap \mathsf{S}^{q\text{-query}}]$ secure hash function H and generates $hk \xleftarrow{\$} \mathsf{H.KGen}(1^\lambda)$. Next it chooses an adaptive NIZK $\mathcal{ZK} = (\mathcal{P}, \mathcal{V})$ and generates its common random string $\text{CRS} \leftarrow \{0,1\}^{\mathsf{Poly}(\lambda)}$. Finally, it invokes $\mathcal{PE.G}(1^\lambda)$ to generate a pair of public key and private key $(pk, sk) \leftarrow \mathcal{PE.G}(1^\lambda)$ and specifies the ciphertext space $\text{CiphS}_{\mathcal{ME}}$ as a tuple

$$((G, G), \{0,1\}^{l(\lambda)}, \{0,1\}^{\mathcal{SE}.CL(\lambda)}, \text{CiphS}_{\mathcal{PE}}).$$

$\mathcal{SE}.CL(\lambda)$ is the ciphertext bit-length of the symmetric encryption $\mathcal{SE}$ and $\text{CiphS}_{\mathcal{PE}}$ is the ciphertext space of the public key encryption $\mathcal{PE}$. At the end, it publishes

$$pp = ((G, G_T, p, g, e), hk, \text{CRS}, pk).$$

While the secret key sk is preserved by the server.

Key derivation. $\mathcal{ME.KD}(pp, m)$ is run by the user. On inputting the public parameter pp and a plaintext m, it computes the key $K = \mathsf{H.Eval}(1^\lambda, hk, m, 1^{l(\lambda)})$ and returns K. Where $m \in \{0,1\}^{\mathcal{ME.IL}(\lambda)}$ and has min-entropy $-\log(\mathsf{Guess}_{\mathcal{M}})$.

Encryption. $\mathcal{ME.E}(pp, K, m)$ is executed by a user. On inputting pp, K and a plaintext $m \in \{0,1\}^{\mathcal{ME.IL}(\lambda)}$, it first chooses two random numbers $r \xleftarrow{\$} Z_p^*$, $L \xleftarrow{\$} \{0,1\}^{l(\lambda)}$ and then computes

$$\tau = (g^r, g^{r\mathsf{H.Eval}(1^\lambda, hk, K, 1^{l_p(\lambda)})}), \quad c_1 = L \oplus K, \quad \kappa = \mathsf{H.Eval}(1^\lambda, hk, L, 1^{l_\kappa(\lambda)}),$$

$$t = \mathsf{H.Eval}(1^\lambda, hk, \tau||c_1||K||L||m, 1^{l_t(\lambda)}), \quad c_2 = \mathcal{SE.E}_\kappa(t||m).$$

Next, it computes $\pi \leftarrow \mathcal{P}(x, w)$ and $v \leftarrow \mathcal{PE.E}(pk, \pi)$, where $x = (\tau, c_1, c_2)$ and $w = (m, r, L)$. Finally the algorithm returns $\tau||c_1||c_2||v$.

Tag generation. $\mathcal{ME.TG}(pp, \tau||c_1||c_2||v, sk)$ is executed by the server which holds the secret key sk. Given sk and $\tau||c_1||c_2||v$, the algorithm first parses $\tau||c_1||c_2||v$ into τ, c_1, c_2 and v, then computes $\pi \leftarrow \mathcal{PE.D}(sk, v)$ and checks whether $\mathcal{V}(x, \pi) = 1$, if so, it sets $T = \tau$, otherwise it outputs reject. Where $x = (\tau, c_1, c_2)$.

Decryption. $\mathcal{ME.D}(pp, K, \tau||c_1||c_2||v)$ is performed by the user. On inputting pp, K and ciphertext $\tau||c_1||c_2||v$, it first parses $\tau||c_1||c_2||v$ into τ, c_1, c_2 and v, then computes

$$L = c_1 \oplus K, \quad \kappa = \mathsf{H.Eval}(1^\lambda, hk, L, 1^{l_\kappa(\lambda)}), \quad t'||m' \leftarrow \mathcal{SE.D}_\kappa(c_2).$$

Next it checks whether $\mathsf{H.Eval}(1^\lambda, hk, \tau||c_1||K||L||m', 1^{l_t(\lambda)}) = t'$. If this is the case, it returns m', otherwise, it returns $\perp$. Where $x = (\tau, c_1, c_2)$.

Equality testing. $\mathcal{ME.ET}(pp, T_1, T_2)$ is run by the server. Once input pp and two tags T_1 and T_2, the algorithm first parses T_1 and T_2 into (τ_{11}, τ_{12}) and (τ_{21}, τ_{22}). Then it checks whether $e(\tau_{11}, \tau_{22}) = e(\tau_{12}, \tau_{21})$, if so, it outputs true, otherwise it outputs false.

The language $\mathcal{L}$ and the relation $\mathcal{R}$. Generally speaking, the language $\mathcal{L}$ consists of statements $x = (\tau, c_1, c_2)$ in which each components are generated with the unknown values $w = (m, r, L)$ in a consistent way. Formally, a relation $\mathcal{R} = \{(x, \omega)\}$, which consists of statements x and its witness ω, is defined as follows. In addition, the language $\mathcal{L}$ must satisfy the form $\mathcal{L} = \{x : \exists\ \omega \text{ s.t. } (x, w) \in \mathcal{R}\}$.

$$\mathcal{R} := \left\{ ((\tau, c_1, c_2), (m, r, L)) : \begin{array}{l} K = \mathsf{H.Eval}(1^\lambda, hk, m, 1^{l(\lambda)}) \\ \tau = (g^r, g^{r\mathsf{H.Eval}(1^\lambda, hk, K, 1^{l_p})}) \\ c_1 = L \oplus K \\ \kappa = \mathsf{H.Eval}(1^\lambda, hk, L, 1^{l_\kappa(\lambda)}) \\ t = \mathsf{H.Eval}(1^\lambda, hk, \tau||c_1||K||L||m, 1^{l_t(\lambda)}) \\ c_2 = \mathcal{SE}.\mathcal{E}_\kappa(t||m) \end{array} \right\}.$$

The correctness of the scheme $\mathcal{ME}$. Given a ciphertext $\tau||c_1||c_2||v$, a key K, a pair (pp, sk), if $(pp, sk) = \mathcal{ME}.\mathcal{PG}(1^\lambda)$, $m \in \{0,1\}^{\mathcal{ME}.\mathcal{IL}(\lambda)}$, $K = \mathsf{H.Eval}(1^\lambda, hk, m, 1^{l(\lambda)})$, $L \in \{0,1\}^{l(\lambda)}$, $r \in Z_p^*$, $c_1 = L \oplus K$, $\tau = (g^r, g^{r\mathsf{H.Eval}(1^\lambda, hk, K, 1^{l_p(\lambda)})})$, $t = \mathsf{H.Eval}(1^\lambda, hk, \tau||c_1||K||L||m, 1^{l_t(\lambda)})$, $\kappa = \mathsf{H.Eval}(1^\lambda, hk, L, 1^{l_\kappa(\lambda)})$, $c_2 = \mathcal{SE}.\mathcal{E}_k(t||m)$ and $v = \mathcal{PE}.\mathcal{E}(pk, \pi)$, we have $\pi = \mathcal{PE}.\mathcal{D}(sk, v)$, $\mathcal{V}(x, \pi) = 1$ and $t||m \leftarrow \mathcal{SE}.\mathcal{D}_k(c_2)$ as required with $x = (\tau, c_1, c_2)$. In addition, if $T_1 = (\tau_{11}, \tau_{12})$ and $T_2 = (\tau_{21}, \tau_{22})$ is created with the same value m, then $e(\tau_{11}, \tau_{22}) = e(\tau_{12}, \tau_{21})$ as required. Thus, the correctness of $\mathcal{ME}$ follows from the above.

The validity-testing for ciphertexts. The validity-testing for ciphertexts proceeds as follows. In the encryption process, we first compute $t = \mathsf{H.Eval}(1^\lambda, hk, \tau||c_1||K||L||m, 1^{l_t(\lambda)})$ by letting the hash function H act on the values τ, c_1, K, L and m and then encrypt $t||m$ using the symmetric encryption $\mathcal{SE}$ to get $c_2 = \mathcal{SE}.\mathcal{E}_k(t||m)$. In the decryption process, we first obtain the value $t'||m'$ by decrypting c_2 with the key κ and then recompute the value $t = \mathsf{H.Eval}(1^\lambda, hk, \tau||c_1||K||L||m', 1^{l_t(\lambda)})$, finally test the validity of the ciphertext by checking whether the equation $t = t'$ holds.

4 Security

The security for the scheme $\mathcal{ME}$ is guaranteed by Theorem 1 and Theorem 2 which are shown as follows.

Theorem 1. Let $\mathcal{ZK}$ be an adaptive NIZK, H be a $\mathsf{UCE}[\mathsf{S}^{\sup} \cap \mathsf{S}^{q\text{-query}}]$ secure hash function, $\mathcal{SE}$ be a ROR secure symmetric encryption scheme, $\mathcal{PE}$ be an IND-CPA secure PKE scheme, then the randomized message-locked encryption $\mathcal{ME}$ is $\mathsf{PRV\text{-}CDA3}$ secure. Particularly, if for all PPT adversary A (excludes the server because we assume it is semi-trusted) and all PPT message distribution $\mathcal{M}$ with guessing probability $\mathsf{Guess}_\mathcal{M}$, there exists PPT adversaries A_0, $\mathsf{S}^{\sup} \cap \mathsf{S}^{q\text{-query}}$, D, B' and C' such that

$$\mathsf{Adv}^{\mathsf{PRV\text{-}CDA3}}_{\mathcal{ME},\mathcal{M},A} \leqslant 2\mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{ZK},A_0}(\lambda) + 2\mathsf{Adv}^{\mathsf{UCE}}_{\mathsf{H},\mathsf{S}^{\sup}\cap\mathsf{S}^{q\text{-query}},\mathsf{D},q(\lambda)}(\lambda) + \mathsf{Adv}^{\mathsf{ROR}}_{\mathcal{SE},B'}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{PE},C'}(\lambda),$$

where $q(\lambda)$ is the maximal times that S is allowed to make query to HASH.

Proof. We give the proof of the theorem via a series of games listed below where $\mathsf{G}_1(\lambda)$ denotes the original PRV-CDA3 game. We first show these games and then give the analysis of the individual game hops. We assume that $|\boldsymbol{M}| = q'$ and $q' = \lfloor \frac{q}{4} \rfloor$. Note that the boxed parts denote the difference between two adjacent games.

```
$\underline{\mathsf{G}_1(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
CRS $\leftarrow \{0,1\}^{\mathsf{Poly}(\lambda)}$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
    $\boldsymbol{K}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{M}[i], 1^{l(\lambda)})$;
    $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*$,  $\boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\boldsymbol{K}'[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{K}[i], 1^{l_p})$;
    $\boldsymbol{T}[i] \leftarrow (g^{\boldsymbol{r}[i]}, g^{\boldsymbol{r}[i] \cdot \boldsymbol{K}'[i]})$;
    $\boldsymbol{c}_1[i] \leftarrow \boldsymbol{L}[i] \oplus \boldsymbol{K}[i]$;
    $\kappa[i] = \mathsf{H.Eval}(1^\lambda, hk, L, 1^{l_\kappa(\lambda)})$;
    $\boldsymbol{t}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{K}[i]||\boldsymbol{L}[i]||$
                                                    $\boldsymbol{M}[i], 1^{l_t(\lambda)})$;
    $\boldsymbol{c}_2[i] \xleftarrow{\$} \mathcal{SE}.\mathcal{E}_{\kappa[i]}(\boldsymbol{t}[i]||\boldsymbol{M}[i])$;
    $\pi[i] \leftarrow \mathcal{P}(\mathbf{x}[i], \mathbf{w}[i])$;
    $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
    $//\mathbf{w}[i] = (\boldsymbol{M}[i], \boldsymbol{r}[i], \boldsymbol{L}[i])$;
    $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
    $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{\mathsf{G}_2(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
$\boxed{\mathrm{CRS} \leftarrow \mathsf{Sim}_1(1^\lambda);}$
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boldsymbol{K}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{M}[i], 1^{l(\lambda)})$;
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{K}'[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{K}[i], 1^{l_p})$;
  $\boldsymbol{T}[i] \leftarrow (g^{\boldsymbol{r}[i]}, g^{\boldsymbol{r}[i] \cdot \boldsymbol{K}'[i]})$;
  $\boldsymbol{c}_1[i] \leftarrow \boldsymbol{L}[i] \oplus \boldsymbol{K}[i]$;
  $\kappa[i] = \mathsf{H.Eval}(1^\lambda, hk, L, 1^{l_\kappa(\lambda)})$;
  $\boldsymbol{t}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{K}[i]||\boldsymbol{L}[i]||\boldsymbol{M}[i], 1^{l_t(\lambda)})$;
  $\boldsymbol{c}_2[i] \xleftarrow{\$} \mathcal{SE}.\mathcal{E}_{\kappa[i]}(\boldsymbol{t}[i]||\boldsymbol{M}[i])$;
  $\boxed{\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i]);}$
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{\mathsf{G}_3(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
$\mathrm{CRS} \leftarrow \mathsf{Sim}_1(1^\lambda)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boxed{\boldsymbol{K}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)};}$
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boxed{\boldsymbol{K}'[i] \xleftarrow{\$} Z_p^*;}$
  $\boldsymbol{T}[i] \leftarrow (g^{\boldsymbol{r}[i]}, g^{\boldsymbol{r}[i] \cdot \boldsymbol{K}'[i]})$;
  $\boldsymbol{c}_1[i] \leftarrow \boldsymbol{L}[i] \oplus \boldsymbol{K}[i]$;
  $\boxed{\kappa[i] \xleftarrow{\$} \{0,1\}^{l_\kappa(\lambda)};}$
  $\boxed{\boldsymbol{t}[i] \xleftarrow{\$} \{0,1\}^{l_t(\lambda)};}$
  $\boldsymbol{c}_2[i] \xleftarrow{\$} \mathcal{SE}.\mathcal{E}_{\kappa[i]}(\boldsymbol{t}[i]||\boldsymbol{M}[i])$;
  $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{\mathsf{G}_4(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
$\mathrm{CRS} \leftarrow \mathsf{Sim}_1(1^\lambda)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boldsymbol{K}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{K}'[i] \xleftarrow{\$} Z_p^*$;
  $\boxed{\boldsymbol{T}[i][1] \xleftarrow{\$} G;}$
  $\boxed{\boldsymbol{T}[i][2] \xleftarrow{\$} G;}$
  $\boxed{\boldsymbol{T}[i] \leftarrow (\boldsymbol{T}[i][1], \boldsymbol{T}[i][2]);}$
  $\boxed{\boldsymbol{c}_1[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)};}$
  $\kappa[i] \xleftarrow{\$} \{0,1\}^{l_\kappa(\lambda)}$;
  $\boldsymbol{t}[i] \xleftarrow{\$} \{0,1\}^{l_t(\lambda)}$;
  $\boldsymbol{c}_2[i] \xleftarrow{\$} \mathcal{SE}.\mathcal{E}_{\kappa[i]}(\boldsymbol{t}[i]||\boldsymbol{M}[i])$;
  $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{\mathsf{G}_5(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
$\mathrm{CRS} \leftarrow \mathsf{Sim}_1(1^\lambda)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boldsymbol{K}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{K}'[i] \xleftarrow{\$} Z_p^*$;
  $\boldsymbol{T}[i][1] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i][2] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i] \leftarrow (\boldsymbol{T}[i][1], \boldsymbol{T}[i][2])$;
  $\boldsymbol{c}_1[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\kappa[i] \xleftarrow{\$} \{0,1\}^{l_\kappa(\lambda)}$;
  $\boldsymbol{t}[i] \xleftarrow{\$} \{0,1\}^{l_t(\lambda)}$;
  $\boxed{\boldsymbol{c}_2[i] \xleftarrow{\$} \{0,1\}^{\mathcal{SE}\cdot\mathsf{CL}(\lambda)};}$
  $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{\mathsf{G}_6(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
CRS $\leftarrow \mathsf{Sim}_1(1^\lambda)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boldsymbol{K}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{K}'[i] \xleftarrow{\$} Z_p^*$;
  $\boldsymbol{T}[i][1] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i][2] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i] \leftarrow (\boldsymbol{T}[i][1], \boldsymbol{T}[i][2])$;
  $\boldsymbol{c}_1[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\kappa[i] \xleftarrow{\$} \{0,1\}^{l_\kappa(\lambda)}$;
  $\boldsymbol{t}[i] \xleftarrow{\$} \{0,1\}^{l_t(\lambda)}$;
  $\boldsymbol{c}_2[i] \xleftarrow{\$} \{0,1\}^{\mathcal{SE}\cdot\mathsf{CL}(\lambda)}$;
  $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boxed{\boldsymbol{v}[i] \xleftarrow{\$} \mathrm{CiphS}_{\mathcal{PE}};}$
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{\mathsf{G}_7(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
CRS $\leftarrow \mathsf{Sim}_1(1^\lambda)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boxed{\boldsymbol{K}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{M}[i], 1^{l(\lambda)});}$
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boxed{\boldsymbol{K}'[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{K}[i], 1^{l_p(\lambda)});}$
  $\boldsymbol{T}[i][1] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i][2] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i] \leftarrow (\boldsymbol{T}[i][1], \boldsymbol{T}[i][2])$;
  $\boldsymbol{c}_1[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boxed{\kappa[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{L}[i], 1^{l_\kappa(\lambda)});}$
  $\boxed{\boldsymbol{t}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{K}[i]||\boldsymbol{L}[i]||}$
                                                  $\boxed{\boldsymbol{M}[i], 1^{l_t(\lambda)});}$
  $\boldsymbol{c}_2[i] \xleftarrow{\$} \{0,1\}^{\mathcal{SE}\cdot\mathsf{CL}(\lambda)}$;
  $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boldsymbol{v}[i] \xleftarrow{\$} \mathrm{CiphS}_{\mathcal{PE}}$;
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{\mathsf{G}_8(\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
$\boxed{\text{CRS} \leftarrow \{0,1\}^{\mathsf{Poly}(\lambda)};}$
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boldsymbol{K}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{M}[i], 1^{l(\lambda)})$;
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{K}'[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{K}[i], 1^{l_p(\lambda)})$;
  $\boldsymbol{T}[i][1] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i][2] \xleftarrow{\$} G$;
  $\boldsymbol{T}[i] \leftarrow (\boldsymbol{T}[i][1], \boldsymbol{T}[i][2])$;
  $\boldsymbol{c}_1[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\kappa[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{L}[i], 1^{l_\kappa(\lambda)})$;
  $\boldsymbol{t}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{K}[i]||\boldsymbol{L}[i]||$
                                                  $\boldsymbol{M}[i], 1^{l_t(\lambda)})$;
  $\boldsymbol{c}_2[i] \xleftarrow{\$} \{0,1\}^{\mathcal{SE}\cdot\mathsf{CL}(\lambda)}$;
  $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boldsymbol{v}[i] \xleftarrow{\$} \mathrm{CiphS}_{\mathcal{PE}}$;
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

```
$\underline{A_0(1^\lambda)}$
$(G, G_T, p, g, e) \xleftarrow{\$} \text{GroupGen}(1^\lambda)$;
$(pk, sk) \xleftarrow{\$} \mathcal{PE}.\mathcal{G}(1^\lambda)$;
$hk \xleftarrow{\$} \mathsf{H.KGen}(1^\lambda)$;
Receive CRS as the first-stage input from its challenger;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
    $\boldsymbol{K}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{M}[i], 1^{l(\lambda)})$;
    $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\boldsymbol{K}'[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{K}[i], 1^{l_p})$;
    $\boldsymbol{T}[i] \leftarrow (g^{\boldsymbol{r}[i]}, g^{\boldsymbol{r}[i] \cdot \boldsymbol{K}'[i]})$;
    $\boldsymbol{c}_1[i] \leftarrow \boldsymbol{L}[i] \oplus \boldsymbol{K}[i]$;
    $\kappa[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{L}[i], 1^{l_\kappa(\lambda)})$;
    $\boldsymbol{t}[i] \leftarrow \mathsf{H.Eval}(1^\lambda, hk, \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{K}[i]||\boldsymbol{L}[i]||$
                                   $\boldsymbol{M}[i], 1^{l_t(\lambda)})$;
    $\boldsymbol{c}_2[i] \xleftarrow{\$} \mathcal{SE}.\mathcal{E}_{\kappa[i]}(\boldsymbol{t}[i]||\boldsymbol{M}[i])$;
    Get $\pi[i]$ as second-stage input from its challenger;
    $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
    $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$pp \leftarrow ((G, G_T, p, g, e), hk, \text{CRS}, pk)$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

Figure 4 Algorithm A_0.

G_1 : This game is the original $\mathsf{PRV\text{-}CDA3}$ game where $b = 1$.

G_2 : This game is equivalent to G_1 except that CRS and π are generated by simulators Sim_1 and Sim_2 instead of CRS $\leftarrow \{0,1\}^{\mathsf{Poly}(\lambda)}$ and $\pi[i] \leftarrow \mathcal{P}(\mathbf{x}[i], \mathbf{w}[i])$.

G_3 : This game is the same as game G_2 except that $\boldsymbol{K}[i]$, $\boldsymbol{K}'[i]$, $\kappa[i]$ and $\boldsymbol{t}[i]$ are respectively chosen randomly from $\{0,1\}^{l(\lambda)}$, Z_p^*, $\{0,1\}^{l_\kappa(\lambda)}$ and $\{0,1\}^{l_t(\lambda)}$.

G_4 : This game is consistent with game G_3 except that $\boldsymbol{T}[i]$ and $\boldsymbol{c}_1[i]$ are respectively chosen at random from (G, G) and $\{0,1\}^{l(\lambda)}$.

G_5 : This game is consistent with G_4 with the exception that $\boldsymbol{c}_2[i]$ is randomly chosen from $\{0,1\}^{\mathcal{SE}\cdot\mathsf{CL}(\lambda)}$.

G_6 : This game is consistent with G_5 except that $\boldsymbol{v}[i]$ is randomly chosen from $\text{CiphS}_{\mathcal{PE}}$.

G_7 : In game G_7, instead of choosing $\boldsymbol{K}[i]$, $\boldsymbol{K}'[i]$, $\kappa[i]$ and $\boldsymbol{t}[i]$ randomly, they are computed by the hash function H.

G_8 : This game is consistent with G_7 except that the simulated CRS is replaced back with real CRS.

In game G_8 we are in an identical setting to the $\mathsf{PRV\text{-}CDA3}$ game where $b = 0$. That is, the challenge ciphertext is answered with randomly chosen values independent of the challenge message. Thus, the advantage of an adversary A in the PRV-CDA3 game can be defined as

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{PRV\text{-}CDA3}}_{\mathcal{ME},\mathcal{M},A} &= 2 \cdot \Pr[\mathsf{PRV\text{-}CDA3}^{A}_{\mathcal{ME},\mathcal{M}}(\lambda)] - 1 \\
&= \Pr[\mathsf{PRV\text{-}CDA3}^{A}_{\mathcal{ME},\mathcal{M}}(\lambda)|b=1] + \Pr[\mathsf{PRV\text{-}CDA3}^{A}_{\mathcal{ME},\mathcal{M}}(\lambda)|b=0] - 1 \\
&= \Pr[\mathsf{G}_1(\lambda)] - \Pr[\mathsf{G}_8(\lambda)] \\
&\leqslant \sum_{i=1}^{7} |\Pr[\mathsf{G}_i(\lambda)] - \Pr[\mathsf{G}_{i+1}(\lambda)]|. \qquad (1)
\end{aligned}$$

It remains to show that the individual games are negligible close.

Lemma 1. If $\Pr[\mathsf{G}_1]$ and $\Pr[\mathsf{G}_2]$ respectively denote the probability that the adversary A wins in game G_1 and game G_2 respectively. Then

$$\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2] \leqslant \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{ZK},A_0}(\lambda). \qquad (2)$$

Proof. Assume the adversary A can distinguish G_1 from G_2, then we can construct an adversary A_0 which employs A to distinguish real proof from simulated proof. Assume that A can distinguish G_1 from G_2, then we can use A to construct A_0 who aims to distinguish real proof from simulated proof. Since the advantage of A_0 in distinguishing real/simulated proof is upper bounded by the zero-knowledge property of $\mathcal{ZK}$, thus the advantage of A distinguishing G_1 from G_2 is upper bounded by that of A_0. The construction of A_0 is listed in Figure 4.

Clearly, when the inputs to A_0 are real CRS and π, A's view in the above game is precisely its view in game G_1, when the inputs to A_0 are simulated CRS and proof π, A's view is exactly its view in G_2, thus we have

$$\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2] \leqslant \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{ZK},A_0}(\lambda).$$

```
$\underline{\mathsf{S}^{\mathsf{HASH}}(1^\lambda)}$
$(G, G_T, p, g, e) \xleftarrow{\$} \text{GroupGen}(1^\lambda)$;
$(pk, sk) \xleftarrow{\$} \mathcal{PE}.\mathcal{G}(1^\lambda)$;
CRS $\leftarrow \mathsf{Sim}_1(1^\lambda)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
  $\boldsymbol{K}[i] \leftarrow \mathsf{HASH}(\boldsymbol{M}[i], 1^{l(\lambda)})$;
  $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
  $\boldsymbol{K}'[i] \leftarrow \mathsf{HASH}(\boldsymbol{K}[i], 1^{l_p})$;
  $\boldsymbol{T}[i] \leftarrow (g^{\boldsymbol{r}[i]}, g^{\boldsymbol{r}[i].\boldsymbol{K}'[i]})$
  $\boldsymbol{c}_1[i] \leftarrow \boldsymbol{L}[i] \oplus \boldsymbol{K}[i]$;
  $\kappa[i] \leftarrow \mathsf{HASH}(\boldsymbol{L}[i], 1^{l_\kappa(\lambda)})$;
  $\boldsymbol{t}[i] \leftarrow \mathsf{HASH}(\boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{K}[i]||\boldsymbol{L}[i]||\boldsymbol{M}[i], 1^{l_t(\lambda)})$;
  $\boldsymbol{c}_2[i] \leftarrow \mathcal{SE}.\mathcal{E}_{\kappa[i]}(\boldsymbol{t}[i]||\boldsymbol{M}[i])$;
  $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
  $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
  $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
  $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$L \leftarrow ((G, G_T, p, g, e), \text{CRS}, pk, Z, \boldsymbol{c})$;
Return $L$.

$\underline{\mathsf{D}(1^\lambda, hk, L)}$
$((G, G_T, p, g, e), \text{CRS}, pk, Z, \boldsymbol{c}) \leftarrow L$;
$pp \leftarrow ((G, G_T, p, g, e), hk, \text{CRS}, pk)$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
Return $b'$.
```

Figure 5 Algorithms S and D.

Lemma 2. If $\Pr[\mathsf{G}_2]$ and $\Pr[\mathsf{G}_3]$ respectively denote the probability that adversary A wins in game G_2 and game G_3. Then

$$\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3] \leqslant \mathsf{Adv}^{\mathsf{UCE}}_{\mathsf{H},\mathsf{S}^{\mathrm{sup}} \cap \mathsf{S}^{q\text{-query}},\mathsf{D},q(\lambda)}(\lambda). \tag{3}$$

Proof. Assume that the adversary A can distinguish G_2 from G_3, then we can use A to construct two adversaries S and D which attempt to break the $\mathsf{UCE}[\mathsf{S}^{\mathrm{sup}} \cap \mathsf{S}^{q\text{-query}}]$ security of the hash function H. Recall that S is given a secure parameter λ with access to oracle HASH, outputs a leakage L, and D is given the leakage L and an evaluation key hk of H and outputs a guessing bit b'. The algorithms S and D are described in Figure 5.

Obviously, when the source S accesses to a keyed hash function, the adversary A runs in game G_2, otherwise A runs in game G_3. Finally the distinguisher D uses A to break the $\mathsf{UCE}[\mathsf{S}^{\mathrm{sup}} \cap \mathsf{S}^{q\text{-query}}]$ security of H. Let c denote the challenge bit of game $\mathsf{UCE}^{\mathsf{S},D}_{\mathsf{H},q(\lambda)}(\lambda)$, then we have

$$\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3] = \mathsf{Pr}[\mathsf{UCE}^{\mathsf{S,D}}_{\mathsf{H},q(\lambda)}(\lambda)|c=1] - \mathsf{Pr}[\mathsf{UCE}^{\mathsf{S,D}}_{\mathsf{H},q(\lambda)}(\lambda)|c=0] = \mathsf{Adv}^{\mathsf{UCE}}_{\mathsf{H,S,D},q(\lambda)}(\lambda).$$

It remains to show that S is statistically unpredictable. Consider an arbitrary predictor P', which is given the leakage L and the access to oracle HASH and has to guess one of the strings $\boldsymbol{M}[i]$, $\boldsymbol{K}[i]$, $\boldsymbol{L}[i]$ and $\boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{K}[i]||\boldsymbol{L}[i]||\boldsymbol{M}[i]$. Given $L = ((G, G_T, p, g, e), \text{CRS}, pk, Z, \boldsymbol{c})$, the strings $\boldsymbol{L}[i]$ and $\boldsymbol{K}[i]$ are still uniformly and independently distributed and $\boldsymbol{M}[i]$ still has the min-entropy $-\log \mathsf{Guess}_{\mathcal{M}}$. Besides, since in each encryption, $\kappa[i]$ is uniform and independent, so P' cannot predict $\boldsymbol{M}[i]$ from $\boldsymbol{c}_2[i]$. Again, even if P' knows $\boldsymbol{K}'[i]$, since P' does not know the hash key hk, the probability that $\boldsymbol{K}'[i]$ is predicted is $\frac{1}{2^{l(\lambda)}}$. In addition, because the output of P' contains at most $q(\lambda)$ elements, hence we have

$$\mathsf{Adv}^{\mathsf{stPRED}}_{\mathsf{S,P}'}(\lambda) \leqslant q(\lambda).q'(\lambda)\left(\mathsf{Guess}_{\mathcal{M}}(\lambda) + \frac{2}{2^{l(\lambda)}} + \frac{\mathsf{Guess}_{\mathcal{M}}(\lambda)}{2^{2l(\lambda)}}\right).$$

As for the required query number, we set $|\boldsymbol{M}| = q'(\lambda)$ and $q'(\lambda) = \lfloor \frac{q(\lambda)}{4} \rfloor$ due to that in each encryption, S is only allowed to issue 4 queries to oracle HASH which is determined by the construction of H. So we have

$$\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3] \leqslant \mathsf{Adv}^{\mathsf{UCE}}_{\mathsf{H},\mathsf{S}^{\mathrm{sup}} \cap \mathsf{S}^{q\text{-query}},\mathsf{D},q(\lambda)}(\lambda).$$

Lemma 3. Let $\Pr[\mathsf{G}_3]$ and $\Pr[\mathsf{G}_4]$ respectively denote the probability that the adversary A wins in game G_3 and game G_4. Then

$$\Pr[\mathsf{G}_3] = \Pr[\mathsf{G}_4]. \tag{4}$$

Proof. The difference between G_3 and G_4 is that in G_3, $\boldsymbol{T}[i]$ and $\boldsymbol{c}_1[i]$ are computed by the encryption algorithm, while in G_4 they are chosen randomly from (G, G) and $\{0,1\}^{l(\lambda)}$ respectively. In fact, $\boldsymbol{T}[i]$ and $\boldsymbol{c}_1[i]$ in G_3 are identically distributed as in G_4 since $\boldsymbol{r}[i]$ and $\boldsymbol{L}[i]$ used to generate $\boldsymbol{T}[i]$ and $\boldsymbol{c}_1[i]$ in G_3 are chosen randomly from Z_p^* and $\{0,1\}^{l(\lambda)}$ respectively. Thus, we have

$$\Pr[\mathsf{G}_3] = \Pr[\mathsf{G}_4].$$

```
$\underline{B'^{\mathsf{Enc1}}(1^\lambda)}$
$(pp, sk) \leftarrow \mathcal{PG}(1^\lambda)$;
$\mathrm{CRS} \leftarrow \mathsf{Sim}_1(1^\lambda)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
    $\boldsymbol{K}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*,\ \boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\boldsymbol{K}'[i] \xleftarrow{\$} Z_p^*$;
    $\boldsymbol{T}[i][1] \xleftarrow{\$} G$;
    $\boldsymbol{T}[i][2] \xleftarrow{\$} G$;
    $\boldsymbol{T}[i] \leftarrow (\boldsymbol{T}[i][1], \boldsymbol{T}[i][2])$;
    $\boldsymbol{c}_1[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\boldsymbol{t}[i] \xleftarrow{\$} \{0,1\}^{l_t(\lambda)}$;
    $\boldsymbol{c}_2[i] \leftarrow \mathsf{Enc1}(\boldsymbol{M}[i])$;
    $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
    $//\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
    $\boldsymbol{v}[i] \xleftarrow{\$} \mathcal{PE}.\mathcal{E}(pk, \pi[i])$;
    $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \boldsymbol{c}, Z)$;
```

Figure 6 Algorithm B'.

Lemma 4. Let $\Pr[\mathsf{G}_4]$ and $\Pr[\mathsf{G}_5]$ respectively denote the probability that the adversary A wins in game G_4 and game G_5. Then

$$\Pr[\mathsf{G}_4] - \Pr[\mathsf{G}_5] \leqslant \mathsf{Adv}^{\mathsf{ROR}}_{\mathcal{SE},B'}(\lambda). \tag{5}$$

Proof. This time, we wish to use the adversary A, which aims to distinguish G_4 from G_5, to create an algorithm B' which attempts to breach the ROR security of $\mathcal{SE}$. Namely, if A can distinguish G_4 from G_5, then B' can distinguish a ciphertext generated by encryption algorithm from a random string. To achieve this goal, B' first uses A to obtain a message distribution $\mathcal{M}$, then gets the ciphertext component of the symmetric encryption $\mathcal{SE}$ from its encryption oracle $\mathsf{Enc1}$, and finally gives this ciphertext together with other ciphertext components generated by itself to A as the challenge ciphertext. We observe that the advantage of B' is exactly that A distinguishes G_4 from G_5. The construction of B' is shown in Figure 6.

In algorithm B', $\boldsymbol{c}_2[i]$ is either a ciphertext generated by encryption algorithm or a random string. When $\boldsymbol{c}_2[i]$ is the former, B' simulates G_4 for A, otherwise, it simulates G_5 for A. Let c be the challenge bit in game $\mathsf{ROR}^{B'}_{\mathcal{SE}}$. Therefore, the advantage that A distinguishes G_4 from G_5 can be bounded by that B' breaks the ROR security of $\mathcal{SE}$. Hence we have

$$\Pr[\mathsf{G}_4] - \Pr[\mathsf{G}_5] = \Pr[\mathsf{ROR}^{B'}_{\mathcal{SE}}(\lambda)|c = 1] - \Pr[\mathsf{ROR}^{B'}_{\mathcal{SE}}(\lambda)|c = 0] \leqslant \mathsf{Adv}^{\mathsf{ROR}}_{\mathcal{SE},B'}(\lambda).$$

Lemma 5. Let $\Pr[\mathsf{G}_5]$ and $\Pr[\mathsf{G}_6]$ respectively denote the probability that the adversary A wins in game G_5 and game G_6. Then

$$\Pr[\mathsf{G}_5] - \Pr[\mathsf{G}_6] \leqslant \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{PE},C'}(\lambda). \tag{6}$$

Proof. Assume that an adversary A can distinguish G_5 from G_6, an algorithm C' can be constructed using A to break the IND-CPA security of $\mathcal{PE}$. Specifically, C' is to distinguish a ciphertext generated by encryption algorithm from that chosen randomly from $\mathrm{CiphS}_{\mathcal{PE}}$. To achieve this goal, we construct C' who tries to use A to break the scheme $\mathcal{PE}$. The algorithm C' is presented in Figure 7.

In algorithm C', when $\boldsymbol{v}[i]$ is a ciphertext generated by encrypting $\boldsymbol{M}[i]$, C' simulates the game G_5 for A, otherwise, when $\boldsymbol{v}[i]$ is a ciphertext generated by encrypting $\mathbf{M}'[i]$, it simulates the setting in game G_6 for A. Hence, the advantage that A distinguishes G_5 from G_6 can be upper bounded by that of C' breaking the IND-CPA security of $\mathcal{PE}$. Let c be the challenge bit in game $\mathsf{IND\text{-}CPA}_{\mathcal{PE}}$, then we have

$$\Pr[\mathsf{G}_5] - \Pr[\mathsf{G}_6] = \Pr[\mathsf{IND\text{-}CPA}^{C'}_{\mathcal{PE}}(\lambda)|c = 1] - \Pr[\mathsf{IND\text{-}CPA}^{C'}_{\mathcal{PE}}(\lambda)|c = 0] \leqslant \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{PE},C'}(\lambda).$$

Lemma 6. Let $\Pr[\mathsf{G}_6]$ and $\Pr[\mathsf{G}_7]$ respectively denote the probability that the adversary A wins in game G_6 and game G_7. Then

$$\Pr[\mathsf{G}_6] - \Pr[\mathsf{G}_7] \leqslant \mathsf{Adv}^{\mathsf{UCE}}_{\mathsf{H},\mathsf{S}^{\mathrm{sup}} \cap \mathsf{S}^{q\text{-query}},\mathsf{D},q(\lambda)}(\lambda). \tag{7}$$

Proof. Since the analysis and results of this Lemma are the same as that of Lemma 2, so here we omit the details about its proof.

Lemma 7. Let $\Pr[\mathsf{G}_7]$ and $\Pr[\mathsf{G}_8]$ respectively denote the probability that the adversary A wins in game G_7 and game G_8. Then

$$\Pr[\mathsf{G}_7] - \Pr[\mathsf{G}_8] \leqslant \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{ZK},A_0}(\lambda). \tag{8}$$

```
$\underline{C'^{\mathsf{Enc2}}(1^\lambda)}$
$(G, G_T, p, g, e) \xleftarrow{\$} \text{GroupGen}(1^\lambda)$;
$hk \xleftarrow{\$} \mathsf{H.KGen}(1^\lambda)$;
Get $pk$ from its challenger;
$\text{CRS} \leftarrow \mathsf{Sim}_1(1^\lambda)$;
$pp \leftarrow ((G, G_T, p, g, e), hk, \text{CRS}, pk)$;
$(\boldsymbol{M}, Z) \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
$(\mathbf{M}', Z') \xleftarrow{\$} \mathcal{M}(1^\lambda)$;
For $i = 1, \ldots, |\boldsymbol{M}|$ do
    $\boldsymbol{K}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\boldsymbol{r}[i] \xleftarrow{\$} Z_p^*$, $\boldsymbol{L}[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\boldsymbol{K}'[i] \xleftarrow{\$} Z_p^*$;
    $\boldsymbol{T}[i][1] \xleftarrow{\$} G$;
    $\boldsymbol{T}[i][2] \xleftarrow{\$} G$;
    $\boldsymbol{T}[i] \leftarrow (\boldsymbol{T}[i][1], \boldsymbol{T}[i][2])$;
    $\boldsymbol{c}_1[i] \xleftarrow{\$} \{0,1\}^{l(\lambda)}$;
    $\kappa[i] \xleftarrow{\$} \{0,1\}^{l_\kappa(\lambda)}$;
    $\boldsymbol{t}[i] \xleftarrow{\$} \{0,1\}^{l_t(\lambda)}$;
    $\boldsymbol{c}_2[i] \xleftarrow{\$} \{0,1\}^{\mathcal{SE}\cdot\mathsf{CL}(\lambda)}$;
    $\pi[i] \xleftarrow{\$} \mathsf{Sim}_2(\mathbf{x}[i])$;
    // $\mathbf{x}[i] = (\boldsymbol{T}[i], \boldsymbol{c}_1[i], \boldsymbol{c}_2[i])$;
    $\boldsymbol{v}[i] \leftarrow \mathsf{Enc2}(\boldsymbol{M}[i], \mathbf{M}'[i])$;
    $\boldsymbol{c}[i] \leftarrow \boldsymbol{T}[i]||\boldsymbol{c}_1[i]||\boldsymbol{c}_2[i]||\boldsymbol{v}[i]$;
$b' \xleftarrow{\$} A(1^\lambda, pp, \mathbf{c}, (Z, Z'))$;
```

Figure 7 Algorithm C'.

```
$\underline{E'(1^\lambda)}$
$(G, G_T, p, g, e) \xleftarrow{\$} \text{GroupGen}(1^\lambda)$;
$(pk, sk) \xleftarrow{\$} \mathcal{PE}.\mathcal{G}(1^\lambda)$;
$hk \xleftarrow{\$} \mathsf{H.KGen}(1^\lambda)$;
Get CRS from its challenger;
$pp \leftarrow ((G, G_T, p, g, e), hk, \text{CRS}, pk)$;
$(M, c') \xleftarrow{\$} A(pp)$;
$(\tau', c_1', c_2', v') \leftarrow c'$;
$T' \leftarrow \tau'$;
$\pi' \leftarrow \mathcal{PE}.\mathcal{D}(sk, v')$;
$M' \leftarrow \mathcal{ME}.\mathcal{D}(pp, \mathcal{ME}.\mathcal{KD}(pp, M), c')$;
$c \leftarrow \mathcal{ME}.\mathcal{E}(pp, \mathcal{ME}.\mathcal{KD}(pp, M), M)$;
$T \leftarrow \mathcal{ME}.\mathcal{TG}(pp, c, sk)$;
If $(\mathcal{V}((\tau', c_1', c_2'), \pi') = 1) \wedge (M' \neq M) \wedge (\mathcal{ME}.\mathcal{ET}(pp, T', T) = \text{true})$
    Return $(((\tau', c_1', c_2'), \pi') \wedge (\tau', c_1', c_2') \notin \mathcal{L})$.
```

Figure 8 Algorithm E'.

Proof. It is easy to see that the probability that A distinguishes G_7 from G_8 is at most $\mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{ZK},\mathsf{P},\mathsf{V},A_0}(\lambda)$ due to the zero-knowledge property of $\mathcal{ZK}$. Thus, we have $\Pr[\mathsf{G}_7] - \Pr[\mathsf{G}_8] \leqslant \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{ZK},A_0}(\lambda)$.

Combining Eqs. (1)–(8), we get

$$\begin{aligned}\mathsf{Adv}^{\mathsf{PRV\text{-}CDA3}}_{\mathcal{ME},\mathcal{M},A} &\leqslant 2\mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{ZK},A_0}(\lambda) + 2\mathsf{Adv}^{\mathsf{UCE}}_{\mathsf{H},\mathsf{S}^{\mathrm{sup}}\cap\mathsf{S}^{q\text{-query}},\mathsf{D},q(\lambda)}(\lambda)\\ &+\mathsf{Adv}^{\mathsf{ROR}}_{\mathcal{SE},B'}(\lambda) + \mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{PE},C'}(\lambda).\end{aligned}$$

Thus, the proof of Theorem 1 is completed.

Theorem 2. Let $\mathcal{ZK}$ be an adaptive NIZK, then the construction for randomized message-locked encryption scheme $\mathcal{ME}$ is strong tag consistency (STC) secure. Particularly, if for all PPT adversary A (Assume A is the semi-trusted server), there exists a PPT algorithm E' such that

$$\mathsf{Adv}^{\mathsf{STC}}_{\mathcal{ME},A}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{soud}}_{\mathcal{ZK},E'}(\lambda) + \frac{1}{2^{l(\lambda)}} + \frac{1}{2^{l_p(\lambda)}}.$$

Proof. First, we give an analysis of the correctness of the tag. Since a semi-trusted server, who preserves the secret key sk, is responsible for running the tag generation algorithm $\mathcal{ME}.\mathcal{TG}(pp, \tau||c_1||c_2||v, sk)$, it can recover the proof $\pi = \mathcal{PE}.\mathcal{D}(sk, v)$ from v. Moreover, if the recovered π is generated correctly for a statement $x = (\tau, c_1, c_2) \in \mathcal{L}$, there must be $\mathcal{V}(\pi, x) = 1$ and thus the server gets a correct tag $T = \tau$.

Next, we prove the strong tag consistency (STC). Assume the adversary A can break the STC security of $\mathcal{ME}$ with probability ϵ, namely $\mathsf{Adv}^{\mathsf{STC}}_{\mathcal{ME},A}(\lambda) = \epsilon$, then we can specify an algorithm E' which can break the soundness of $\mathcal{ZK}$ with probability at least $\epsilon - \frac{1}{2^{l(\lambda)}} - \frac{1}{2^{l_p(\lambda)}}$. The construction of E' is shown in Figure 8.

We can see that E' perfectly simulates the attacking environment for A. Note that if the conditional statement "If" is true, then E' successfully outputs a pair $((\tau', c_1', c_2'), \pi')$ with $(\tau', c_1', c_2') \notin \mathcal{L}$ to make $\mathcal{V}((\tau', c_1', c_2'), \pi') = 1$ true and thus breaks the soundness security of $\mathcal{ZK}$. Moreover, also note that the probability that the adversary A successfully forges a tag is just $\Pr[\mathcal{V}((\tau', c_1', c_2'), \pi') = 1, M' \neq$

$M, \mathcal{ME}.\mathcal{ET}(pp, T', T) =$ true]. In the following, we analyze this probability. In order to facilitate the writing, the following events are defined:

E1: when the event "$\mathcal{V}((\tau', c_1', c_2'), \pi') = 1$" happens;

E2: when the event " $M' \neq M$" happens;

E3: when the event "$\mathcal{ME}.\mathcal{ET}(pp, T', T) =$ true" happens.

Now, we can write $\Pr[\mathcal{V}((\tau', c_1', c_2'), \pi') = 1, M' \neq M, \mathcal{ME}.\mathcal{ET}(pp, T', T) = \text{true}]$ as $\Pr[\text{E1}, \text{E2}, \text{E3}]$. Furthermore, in terms of $(\tau', c_1', c_2') \in \mathcal{L}$ or $(\tau', c_1', c_2') \notin \mathcal{L}$, the probability $\Pr[\text{E1}, \text{E2}, \text{E3}]$ can be written as

$$\Pr[\text{E1}, \text{E2}, \text{E3}] \leqslant \Pr[\text{E1}, \text{E2}, \text{E3} | (\tau', c_1', c_2') \in \mathcal{L}] + \Pr[\text{E1}, \text{E2}, \text{E3}, (\tau', c_1', c_2') \notin \mathcal{L}].$$

First, we analyze the probability $\Pr[\text{E1}, \text{E2}, \text{E3} | (\tau', c_1', c_2') \in \mathcal{L}]$. Conditioned on $(\tau', c_1', c_2') \in \mathcal{L}$, we can get: (1) If E1 happens, this means that T' has the form like $(g^{r'}, g^{r'.\mathsf{H.Eval}(1^\lambda, hk, K', 1^{l_p})})$ for $r' \in Z_p^*$. (2) If E3 happens, we have $e(g^{r'}, g^{r.\mathsf{H.Eval}(1^\lambda, hk, K, 1^{l_p})}) = (g^{r'.\mathsf{H.Eval}(1^\lambda, hk, K', 1^{l_p})}, g^r)$, namely $e(g, g)^{r'r.\mathsf{H.Eval}(1^\lambda, hk, K, 1^{l_p})}$ $= (g, g)^{r'r.\mathsf{H.Eval}(1^\lambda, hk, K', 1^{l_p})}$. Therefore, when $(\tau', c_1', c_2') \in \mathcal{L}$, with (1), (2) and E2 together, there must be

$$\mathsf{H.Eval}(1^\lambda, hk, \mathsf{H.Eval}(1^\lambda, hk, M, 1^{l(\lambda)}), 1^{l_p}) = \mathsf{H.Eval}(1^\lambda, hk, \mathsf{H.Eval}(1^\lambda, hk, M', 1^{l(\lambda)}), 1^{l_p}).$$

Furthermore, since the output of the hash function H is uniformly and independently distributed, the probability $\Pr[\text{E1}, \text{E2}, \text{E3} | (\tau', c_1', c_2') \in \mathcal{L}]$ can be upper bounded by $\frac{1}{2^{l(\lambda)}} + \frac{1}{2^{l_p(\lambda)}}$.

Next, we analyze the probability $\Pr[\text{E1}, \text{E2}, \text{E3}, (\tau', c_1', c_2') \notin \mathcal{L}]$. Note that

$$\Pr[\text{E1}, \text{E2}, \text{E3}, (\tau', c_1', c_2') \notin \mathcal{L}] \leqslant \Pr[E1, (\tau', c_1', c_2') \notin \mathcal{L}],$$

while the probability $\Pr[\text{E1}, (\tau', c_1', c_2') \notin \mathcal{L}]$ is just that E' broke the soundness of $\mathcal{ZK}$. Thereby $\Pr[\text{E1}, \text{E2}, \text{E3}, (\tau', c_1', c_2') \notin \mathcal{L}] \leqslant \mathsf{Adv}^{\mathsf{soud}}_{\mathcal{ZK}, E'}(\lambda)$. Hence we have

$$\begin{aligned}
\Pr[\text{E1}, \text{E2}, \text{E3}] &\leqslant \Pr[\text{E1}, \text{E2}, \text{E3} | (\tau', c_1', c_2') \in \mathcal{L}] + \Pr[\text{E1}, \text{E2}, \text{E3}, (\tau', c_1', c_2') \notin \mathcal{L}] \\
&\leqslant \Pr[\text{E1}, \text{E2}, \text{E3} | (\tau', c_1', c_2') \in \mathcal{L}] + \Pr[E1, (\tau', c_1', c_2') \notin \mathcal{L}] \\
&\leqslant \frac{1}{2^{l(\lambda)}} + \frac{1}{2^{l_p(\lambda)}} + \mathsf{Adv}^{\mathsf{soud}}_{\mathcal{ZK}, E'}(\lambda).
\end{aligned}$$

Namely

$$\mathsf{Adv}^{\mathsf{STC}}_{\mathcal{ME}, A}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{soud}}_{\mathcal{ZK}, E'}(\lambda) + \frac{1}{2^{l(\lambda)}} + \frac{1}{2^{l_p(\lambda)}}.$$

Hence, the proof of Theorem 2 is completed.

Remark 3. In scheme $\mathcal{ME}$, if we eliminate the encryption of proof π, then we can implement a privacy chosen-distribution attacks (PRV-CDA) and strong tag consistency (STC) secure randomized message-locked encryption scheme (denoted as $\mathcal{ME}'$) in the standard model via UCEs. Furthermore, proofs about privacy chosen-distribution attacks (PRV-CDA) and strong tag consistency (STC) securities of $\mathcal{ME}'$ can be completed by combining the techniques in this paper and that in Naor-Yung scheme for indistinguishable chosen-ciphertext attacks (IND-CCA) security [27].

5 Comparison with related work

In the following, we give a comparison between our scheme and other randomized message-locked encryptions in security. Since our work mainly focuses on the improvements on the security, we do not consider the performance and complexity here for our scheme.

Security comparison. Here we show some comparisons among our scheme and other randomized message-locked encryptions in security model and the security notions: PRV-CDA, PRV-CDA2, PRV-CDA3, TC and STC. Obviously, the notion of PRV-CDA3 is stronger than that of PRV-CDA. However, since PRV-CDA2 requires that the message distribution may depend on the public parameter and be (T, k)-block-source or (T, k)-source, it is hard to find some measures to make comparison between PRV-CDA2 and PRV-CDA (or between PRV-CDA2 and PRV-CDA3). Abadi et al. claimed that their randomized MLE [16] was k-source PRV-CDA2 secure in the random oracle model under the min-entropy DDH

Table 1 Security model, PRV-CDA, PRV-CDA2, PRV-CDA3, TC and STC

Schemes	Security model	PRV-CDA	PRV-CDA2 [16]	PRV-CDA3	TC	STC
RCE [1]	RO	√	×	×	×	×
XtESPKE [1]	STD	√	×	×	√	√
Abadi et al.'s [16]	RO	√	√	−	√	√
$\mathcal{ME}'$	STD	√	×	×	√	√
$\mathcal{ME}$	STD	√	–	√	√	√

and CDH assumptions, but they did not give any rigorous proofs. However, we still make a comparison between Abadi et al.'s scheme and other schemes. Bellare et al. pointed out that their XtESPKE scheme in [1] could be obtained by combining an efficiently-searchable public key encryption and a particular hash function and declared that their scheme achieved both PRV-CDA and STC security, but they did not give specific construction and proofs. Likewise, we consider the XtESPKE scheme in the comparison as well. We summarize these results in Table 1. Note that the notation "–" in the table means unknown securities.

From Table 1, we can see that our scheme $\mathcal{ME}$ achieves better securities in data privacy and tag consistency than RCE, namely, $\mathcal{ME}$ achieves PRV-CDA3 and STC attacks, while RCE only obtains PRV-CDA and TC securities. Compared with Abadi et al.'s scheme, which only achieves PRV-CDA2 and STC securities in the random oracle model, our scheme $\mathcal{ME}$ obtains PRV-CDA3 and STC securities in the standard model and provides formal reductions and proofs. Relative to scheme XtESPKE and $\mathcal{ME}'$, both of which merely achieve PRV-CDA security in data privacy, our scheme $\mathcal{ME}$ not only captures stronger PRV-CDA3 security via blackbox reductions, but also provides detailed security proofs. In fact, the schemes XtESPKE and $\mathcal{ME}'$ have the same-level securities in data privacy and tag consistency, so the latter may be seen as a new way of implementing the former. In addition, although the interactive message-locked encryption scheme proposed by Bellare et al. [17] is also a randomized MLE, since it needs too many interactions, we do not include it in the comparison.

6 Conclusion remarks

We present a new variant of randomized message-locked encryption, MLE3, and define a new security model, PRV-CDA3, for it. Meanwhile, we give a concrete construction and prove that it can achieve the strong tag consistency securities (STC) and privacy chosen-distribution attacks3 (PRV-CDA3) securities in the standard model via blackbox reductions. Compared with existing randomized message-locked encryptions, the new scheme $\mathcal{ME}$ has more advantages in security. In addition, the open problem in this paper is how to design randomized message-locked encryptions which can achieve both STC and PRV$-CDA securities in the standard model.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. NSFC61133014, NSFC61472114, NSFC61572318, NSFC61272440, NSFC61472251, U1536101, NSFC61502400), Foundation of Sichuan Educational Committee (Grant No. 16ZB0140) and Natural Science Foundation of Anhui Science and Technology University (Grant No. ZRC2013380).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Bellare M, Keelveedhi S, Ristenpart T. Message-locked encryption and secure deduplication. In: Advances in Cryptology–EUROCRYPT 2013. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2013. 7881: 296–312

2 Xu Z W. Cloud-sea computing systems: towards thousand-fold improvement in performance per watt for the coming zettabyte era. J Comput Sci Tech, 2014, 29: 177–181

3 Zhang T, Ma J F, Li Q, et al. Trust-based service composition in multi-domain environments under time constraint. Sci China Inf Sci, 2014, 57: 092109

4 Douceur J R, Adya A, Bolosky W J, et al. Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, Vienna, 2002. 617–624

5 Adya A, Bolosky W, Castro M, et al. Farsite: federated, available, and reliable storage for an incompletely trusted environment. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation. New York: ACM, 2002. 1–14

6 Anderson P, Zhang L. Fast and secure laptop backups with encrypted de-duplication. In: Proceedings of the 24th International Conference on Large Installation System Administration. Berkeley: USENIX Association, 2010. 1–8

7 Houssem J, Maryline L-M. Pstore: a secure peer-to-peer backup system. In: Proceedings of the 8th International Conference on New Technologies in Distributed Systems. New York: ACM, 2008. 130–139

8 Cooley J, Taylor C, Peacock A. Abs: the apportioned backup system. Proc Csee, 2011, 31: 112–118

9 Cox L P, Murray C D, Noble B D. Pastiche: making backup cheap and easy. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation. New York: ACM, 2002. 36: 285–298

10 Killijian M-O, Courtes L, Powell D. A survey of cooperative backup mechanisms. https://hal.archives-ouvertes.fr/hal-00139690/document. 2006

11 Marques L, Costa C. Secure deduplication on mobile devices. In: Proceedings of the Workshop on Open Source and Design of Communication. New York: ACM, 2011. 19–26

12 Rahumed A, Chen H C H, Tang Y, et al. A secure cloud backup system with assured deletion and versioncontrol. In: Proceedings of the 40th International Conference on Parallel Processing Workshops, Taipei City, 2011. 160–167

13 Storer M, Greenan K, Long D, et al. Secure data deduplication. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability. New York: ACM, 2008. 1–10

14 O'Hearn Z-W, Warner B. Tahoe: the least-authority filesystem. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability. New York: ACM, 2008. 21–26

15 Horng G B. A new method for constructing multiple assignment schemes for generalized secret sharing. J Inf Sci Eng, 2001, 17: 959–965

16 Abadi M, Boneh D, Mironov I, et al. Message-locked encryption for lock-dependent messages. In: Advances in Cryptology–CRYPTO 2013. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2013. 8042: 374–391

17 Bellare M, Keelveedhi S. Interactive message-locked encryption and secure deduplication. In: Public-Key Cryptography–PKC 2015. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2015. 9020: 516–538

18 Canetti R, Goldreich O, Halevi S. The random oracle methodology, revisited. J ACM, 2004, 51: 557–594

19 Bellare M, Hong T, Keelveedhi S. Instantiating random oracle via UCEs. In: Advances in Cryptology–CRYPTO 2013. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2013. 8043: 398–415

20 Brzuska C, Mittelbach A. Using indistinguishability obfuscation via uces. In: Advances in Cryptology–ASIACRYPT 2014. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2014. 8874: 122–141

21 Brzuska C, Farshim P, Mittelbach A. Indistinguishability obfuscation and uces: the case of computationally unpredictable sources. In: Advances in Cryptology–CRYPTO 2014. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2014. 8616: 188–205

22 Bellare M, Rogaway P. The security of triple encryption and a framework for code-based game-playing proofs. In: Advances in Cryptology–EUROCRYPT 2006. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2006. 4004: 409–426

23 Shacham H, Ristenpart T, Shrimpton T. Careful with composition: limitations of the indiferentiability framework. In: Advances in Cryptology–EUROCRYPT 2011. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2011. 6632: 487–506

24 Sahai A, Waters B. How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing. New York: ACM, 2014. 475–484

25 Koppula V, Lewko A B, Waters B. Indistinguishability obfuscation for turing machines with unbounded memory. In: Proceedings of the 47th Annual ACM on Symposium on Theory of Computing. New York: ACM, 2015. 419–428

26 Lynn B, Prabhakaran M, Sahai A. Positive results and techniques for obfuscation. In: Advances in Cryptology–EUROCRYPT 2004. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2004. 3027: 20–39

27 Naor M, Yung M. Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the ACM Symposium on the Theory of Computing. New York: ACM, 1990. 427–437