Efficient Fully Secure Leakage-Deterring Encryption

Jan Camenisch[1], Maria Dubovitskaya[1] and Patrick Towa[2,3]

[1] DFINITY*
[2] IBM Research – Zurich
[3] ENS and PSL Research University

Abstract. Encryption is an indispensable tool for securing digital infrastructures as it reduces the problem of protecting the data to just protecting decryption keys. Unfortunately, this also makes it easier for users to share protected data by simply sharing decryption keys.

Kiayias and Tang (ACM CCS 2013) were the first to address this important issue pre-emptively rather than a posteriori like traitor tracing schemes do. They proposed *leakage-deterring* encryption schemes that work as follows. For each user, a piece of secret information valuable to her is embedded into her public key. As long as she does not share her ability to decrypt with someone else, her secret is safe. As soon as she does, her secret is revealed to her beneficiaries. However, their solution suffers from serious drawbacks: (1) their model requires a fully-trusted registration authority that is privy to user secrets; (2) it only captures a CPA-type of privacy for user secrets, which is a very weak guarantee; (3) in their construction which turns any public-key encryption scheme into a leakage-deterring one, the new public keys consist of linearly (in the bit-size of the secrets) many public keys of the original scheme, and the ciphertexts are large.

In this paper, we redefine leakage-deterring schemes. We remove the trust in the authority and guarantee full protection of user secrets under CCA attacks. Furthermore, in our construction, all keys and ciphertexts are short and constant in the size of the secrets. We achieve this by taking a different approach: we require users to periodically refresh their secret keys by running a protocol with a third party. Users do so anonymously, which ensures that they cannot be linked, and that the third party cannot perform selective failure attacks. We then leverage this refresh protocol to allow for the retrieval of user secrets in case they share their decryption capabilities. This refresh protocol also allows for the revocation of user keys and for the protection of user secrets in case of loss or theft of a decryption device. We provide security definitions for our new model as well as efficient instantiations that we prove secure.

Keywords: Accountability · Leakage-Deterring Encryption · Public-Key Encryption

* Work done while the corresponding authors were at IBM Research – Zurich.

1 Introduction

Encryption is a powerful instrument to ensure data confidentiality and to ease data protection: only decryption keys need to be protected. However, it makes sharing protected data much easier as well, just by sharing secret keys only. Users might share decryption keys for different reasons: out of convenience or malice, or accidentally due to poor key management. For example, malicious users might share access to paid services or sell access to company confidential information. Therefore, preventing users from sharing their secret keys, either in the clear or by providing a possibly obfuscated decryption algorithm, is an important problem in deploying cryptographic systems, in particular in corporate environments.

To address this issue, one could use pieces of tamper-proof hardware. However, these are expensive and strenuous to build, deploy and manage. A better approach is a software only solution as proposed by Kiayias and Tang: Leakage-Deterring Encryption (LDE) schemes [18]. The main idea behind LDE schemes is to have an authority embed into each user's public key some valuable secret information that she would rather keep private, and which is revealed as soon as she shares her decryption capabilities. Such secret information could for instance be bitcoin-account secret key [19]. In addition to the secrecy of messages against chosen-ciphertext attacks, an LDE scheme should at least satisfy the following properties:

Privacy of user secrets: provided that a user does not share her decryption capabilities, her secret must not be recoverable from her public key, even if all other parties (including the registration authority which embeds the secrets) are malicious. Ideally, privacy should hold even under CCA attacks.

Recoverability: anyone with a device capable of decrypting, with non-negligible probability, a non-negligible amount of ciphertexts of a user should be able to retrieve her secret.

There precisely lies the complexity of designing LDE schemes: two seemingly antagonistic properties must be bridged, in a context in which users are adversarial and have knowledge of secret information. In cryptography, to overcome apparent paradoxes, one usually assumes the adversary not to be privy to crucial secret information. In the present case, an adversarial user does not only know her secret information but also the device meant to be used against her. She could implement a decryption device that rejects any decryption query that results in the secret, and this rejection would not contradict the fact that her device decrypts a non-negligible amount of ciphertext computed with her public key. Given this observation, to hope to recover user secrets, recovery queries made to decryption devices must be indistinguishable from decryption queries accepted by rogue devices; hence the difficulty to design LDE schemes.

Kiayias and Tang provide schemes [18, 19] that partially satisfy those requirements. Namely, they provide two constructions of LDE schemes which turn any public-key encryption scheme into an LDE scheme. The first construction

requires the original encryption scheme to be additively homomorphic and is efficient, and the second one applies to any encryption scheme (i.e., is generic) but is prohibitively inefficient. Indeed, the public keys of their generic construction of LDE schemes grow linearly (even with the use of error-correcting codes) in the bit-size of the secrets, and the ciphertexts grow with a factor (log of the inverse) that depends on the assumed minimal correctness rate of pirate decryption devices. It already means that if a user designs a pirate decryption device with a decryption success probability that is non-negligibly smaller than the assumed minimal correctness rate though non-negligible, recoverability is not assured. In addition to that, for secrets of sensible length (e.g., 128 bits) and a decryption device with a conceivable correctness rate (e.g., 50%), the scheme is impractical. Moreover, their model only captures a CPA-type of privacy of user secrets: an attacker that can launch a chosen-ciphertext attacks on a user can also recover her secret. Lastly, in their model of LDE schemes, the registration authority is privy to user secrets, which weakens even further user privacy.

1.1 Contributions

We redefine LDE schemes and design a new model that encompasses all the security properties that an LDE scheme should satisfy. The new model captures a CCA-type of privacy of user secrets (even with respect to the registration authority), recoverability and secrecy of messages under CCA attacks:

- We first show that a CCA-type of privacy of user secrets and recoverability can simultaneously be achieved. As noted by Kiayias and Tang, the fact that recovery queries made must be indistinguishable from standard decryption queries implies that a CCA-type of privacy of user secrets and recoverability cannot a priori coexist: any decryption oracle in a privacy security game could be used to perform recovery queries and surely win the game. It comes as a disappointment as it means that an honest user would put her secret at risk under CCA attacks for the sake of protecting the data of a company or of a service provider; data which would withstand those same attacks. A sensible user would clearly not want to use such a system as it does not guarantee the same protection for user secrets as for the encrypted data. Except that this observation is not entirely true: we point out that *a CCA-type of privacy and recoverability can coexist if recovery queries involve a step that cannot be performed with a definition oracle.* We point that since adversaries do not control the randomness of oracles in security definitions, CCA privacy and recoverability[4]can actually be bridged: if recovery queries require to control the randomness of rogue decryption devices, the previous impossibility result does not apply.

[4] Of course, if adversaries were to know the randomness used by security-game oracles, the outputs of those would be deterministic in the view of the adversaries, and even simple properties like IND-CPA would not be satisfiable.

In our construction, the idea is to leverage rewinding to extract secrets from decryption devices. Note that rewinding an algorithm, which solely consists in controlling randomness and not reverse-engineering it, is a black-box technique [2, 4]. Rewinding is used in the context of zero-knowledge proofs to extract witnesses, hence our idea of introducing a third party (which may as well be the embedding authority) that assists users with decryption. To decrypt ciphertexts, users are required to perform a zero-knowledge proof involving their secrets. We prove that any partially functional decryption device needs to do so as well with non-negligible probability, and can therefore be employed to extract secrets.

To reduce communication, we introduce leakage-deterring systems that operate in time periods. We call them time-based leakage-deterring encryption (TB-LDE) schemes. The first time (and only the first time) a user wishes to decrypt a ciphertext in a given time period, she needs to request from the third party a key for the time period by executing a key-derivation protocol. The frequency at which the key-derivation protocol is to be executed (i.e., the length of a time period, e.g., a week) can be adjusted by the system manager. A rogue decryption device that can decrypt ciphertexts in a time period must then also hold the key for it.

Note that a *minimal requirement* to ensure recoverability via interaction while guaranteeing CCA privacy is that the device must be able to decrypt a ciphertext for a future time period, i.e., for which the user does not have the key yet [5]. Indeed, if a user already has the key for a time period, she can hard-code it in the device and avoid any interaction for this period; making the recovery of her secret with ciphertexts encrypted for the period impossible. To ensure recoverability via interaction, users must not always possess all the information required to decrypt. It is an aspect *inherent* to any LDE scheme that leverages interaction to bridge CCA privacy and recoverability. In Section 4, we construct a TB-LDE scheme that satisfies both properties under this minimal requirement.

Despite the introduction of a third party that assists users with decryption, in our model, users remain anonymous and untraceable when they request keys to the third party. The presence of a third party also allows for the revocation of user keys in case of misconduct, or loss or theft of a decryption device. These untraceability and revocation properties are also properly modelled and formally defined in Sections 3 and 8 respectively.

– Secondly, in comparison to Kiayias and Tang's model [18], we do not assume the authority to be privy to the secrets (it only receive commitments), and thus guarantee more privacy. The authority is only used to ensure that the secrets are correctly integrated into the public keys and to vouch for the latter. One may however think that only seeing a commitment prevents the authority from making sure that the secret is indeed valuable to the user and not just garbage. In fact, the same issue already arises when the authority sees the secrets in the clear. If the secret is a bitcoin-account secret key, the

[5] If time periods are short, then any useful device should be able to do so.

authority can check that it is the valid secret key for a given account, but cannot know whether the money is later moved from the account without resorting to a form of tracing scheme (e.g., by regularly checking a public ledger). However, deterring schemes exactly aim to avoid tracing. Moreover, even if the authority were to see secrets in clear, value lies in the eyes of the beholder: the authority cannot tell if the user is concerned about the money in that account, if she has for instance other sources of income. We therefore assume that there is higher-level mechanism that ascertains the value of the secret. In the case of a bitcoin account, it could for example be a mechanism that verifies that the commitment the authority receives is indeed a commitment to the secret key of an account on which the user will receive her *future* salary, assuming that the user cares about it. (Using future salary frees the authority from the need to regularly check a public ledger since the money is yet to come.)

– Lastly, we provide efficient constructions that fulfill all these security requirements. The first time-based construction (in Section 4) is proved secure in the plain model, and the second one (in Section 6), which is even more efficient, is proved secure in the Random Oracle Model (ROM). In Section 7, we give a definition of LDE schemes which satisfy CCA privacy of user secrets and recoverability thanks to interaction, but which require users to interact every time a ciphertext is to be decrypted. Such schemes are relevant when only few ciphertexts are to be decrypted compared to the provided bandwidth, as the burden of interaction would not be prohibitive.
Furthermore, we show in Section 9 how, in combination with revocation, interaction with the key-derivation party in the recovery process can be enforced and leveraged to protect a user's secret and prevent misuse of her legitimate decryption device in case she was simply lost it or was stolen. Those are critical functionalities for deploying such a system.

1.2 Related Work.

Kiayias and Tang were the first to consider typical public-key-infrastructure functionalities such as encryption, signatures and identification [18] in the context of leakage-deterrence. Nevertheless, deterring users from sharing their keys has been prominently considered in the context of broadcast encryption, in which malicious accredited users might implement pirate decoders. In such a scenario, traitor-tracing schemes [13] were introduced. They aim at identifying at least one of possibly many users that colluded to produce a pirate decryption device. Several efficiency improvements have been proposed thereupon [7–9, 20, 21], and variants such as public traitor tracing or anonymous traitor tracing [24] have also been considered. Contrary to traitor-tracing schemes, leakage-deterring schemes follow a proactive approach rather than rely on the identification of malicious users to enforce penalties.

A concept closer to the leakage-deterrence paradigm is that of *self enforcement* [15], which also involves private user pieces of information. In a multi-user encryption system, the adversary controls a set of malicious-user keys, and wishes

to redistribute a plaintext. A self-enforcing schemes ensures that the adversary has to either send a message as long as the plaintext or leak some information about the private information of the traitors. However, recovery in those systems assumes direct access to user keys, i.e., white-box access, and the proposed construction relies on an unfalsifiable assumption.

2 Preliminaries

This section introduces the notation and some building blocks used in the paper.

2.1 Notation

Unless stated otherwise, p is a prime number, and $\mathbb{Z}_p$ denotes the p-order field. For an integer $n \geq 1$, $\mathrm{GL}_n(\mathbb{Z}_p)$ denotes the set of invertible $n \times n$ $\mathbb{Z}_p$-matrices, and $\mathbf{I}_n$ its neutral element. Given an n-dimensional $\mathbb{Z}_p$ vector space $\mathbb{V} \cong \mathbb{Z}_p^n$, $(\mathbf{e}_i)_i$ denotes its canonical basis. For a family $\mathbb{V}_k$ of N_k-dimensional vector spaces, the canonical basis of $\mathbb{V}_k$ is denoted by $(\mathbf{e}_{k,i})_i$. For $g \in \mathbb{G}$, $\mathrm{diag}(g)$ represents the matrix with g on the diagonal and $1_{\mathbb{G}}$ elsewhere. Besides, for $x \in \mathbb{Z}_p$ and $\mathbf{g} \in \mathbb{G}^n$, define $x\mathbf{g} \coloneqq \mathbf{g}^x$. For an integer $k \geq 1$, a matrix $\mathbf{A} \in \mathbb{G}^{n \times k}$ and a vector $\mathbf{x} \in \mathbb{Z}_p^k$, $(\mathbf{A}\mathbf{x})_i = \left(\prod_j \mathbf{x}_j \mathbf{A}_{ij}\right)_i = \left(\prod_j \mathbf{A}_{ij}^{\mathbf{x}_j}\right)_i$. Likewise, for $\mathbf{y} \in \mathbb{Z}_p^n$, $(\mathbf{y}\mathbf{A})_j = \left(\prod_i \mathbf{y}_i \mathbf{A}_{ij}\right)_j = \left(\prod_i \mathbf{A}_{ij}^{\mathbf{y}_i}\right)_j$. To indicate that a random variable X has a distribution $\mathcal{D}$, the notation $X \leftarrow\!\!\$\, \mathcal{D}$ is used. When $\mathcal{D}$ is the uniform distribution over a finite set $\mathcal{X}$, the notation $X \in_R \mathcal{X}$ is used instead. The predictive probability $p(\mathcal{D})$ of a distribution $\mathcal{D}$ is defined as $\max_{x \in \mathcal{X}} p_x$, with p_x being the probability that a $\mathcal{D}$-distributed random variable takes value $x \in \mathcal{X}$. Given a relation $\mathcal{R}$, the notation $\mathsf{PoK}\{w \colon (x, w) \in \mathcal{R}\}$ is used for a Proof of Knowledge (PoK) for the corresponding language; and its extractor is denoted $\mathcal{K}$.

2.2 Pairing Groups and Hardness Assumptions

This section introduces pairing groups and classical hardness assumptions.

Pairing Groups. A pairing group consists of a tuple $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e)$ such that p is a prime number, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are (cyclic) p-order groups, and $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently-computable non-degenerate bilinear map (also called pairing), i.e., $e(P_1, P_2) \neq 1_{\mathbb{G}_T}$, and $\forall a, b \in \mathbb{Z}_p, e(P_1^a, P_2^b) = e(P_1, P_2)^{ab}$. Let $\mathcal{G}_{\mathrm{bpg}}(1^\lambda)$ denote an algorithm that takes as an input a security parameter 1^λ, and outputs the description $\mathcal{PP}_{\mathbb{G}}$ of a pairing group.

***q*-Strong Diffie–Hellman Assumption.** Let $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e)$ be a pairing group. The q-Strong Diffie–Hellman (qSDH) problem [5] in $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e)$ consists in computing a pair $\left(y, P_1^{1/(u+y)}\right) \in \mathbb{Z}_p \backslash \{-u\} \times \mathbb{G}_1$ given a $(p+3)$-tuple $(P_1, P_1^u, P_1^{u^2}, \ldots, P_1^{u^q}, P_2, P_2^u) \in \mathbb{G}_1^{p+1} \times \mathbb{G}_2^2$. The qSDH assumption over $\mathcal{G}_{\mathrm{bpg}}$ is that no efficient algorithm has a non-negligible probability to solve the qSDH problem in $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e) \leftarrow \mathcal{G}_{\mathrm{bpg}}(1^\lambda)$.

Decisional Linear Assumption. The (2-)Decisional Linear (DLIN) assumption [6] over a group generator $\mathcal{G}$ is that for all p-order group $\mathbb{G} = \langle P \rangle \leftarrow \mathcal{G}$, for $a, b, x, y, z \in_R \mathbb{Z}_p$, the distributions of $(P, P^a, P^b, P^{ax}, P^{by}, P^{x+y})$ and $(P, P^a, P^b, P^{ax}, P^{by}, P^z)$ are computationally indistinguishable.

In a pairing group, the symmetrix DLIN assumption is that DLIN holds in $\mathbb{G}_1$ and $\mathbb{G}_2$.

2.3 BBS+ Signatures

The BBS+ signature scheme (as described by Au et al. [1] and inspired by a group signature [6] introduced by Boneh et al.) is a tuple of algorithms $(\mathsf{SignSetup}, \mathsf{SignKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ with

$\mathsf{SignSetup}(1^\lambda) \to \mathcal{PP}$: output $\mathcal{PP} = \mathcal{PP}_{\mathbb{G}}$ the description of a pairing group $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e)$ calling on $\mathcal{G}_{\text{bpg}}(1^\lambda)$

$\mathsf{SignKeyGen}(\mathcal{PP}, n) \to (sk, vk)$: generate $H_0, \ldots, H_n \in_R \mathbb{G}_1, u \in_R \mathbb{Z}_p$, computes $U = P_2^u$, and output $sk = u$, $vk = (U, H_0, \ldots, H_n)$

$\mathsf{Sign}(sk, \mathbf{m} \in \mathbb{Z}_p^n) \to \sigma$: generate $y, z \in_R \mathbb{Z}_p$, compute $V = P_1 H_0^z \prod_{i \geq 1} H_i^{m_i}$ and $W = V^{1/(u+y)}$, and outputs $\sigma = (W, y, z)$

$\mathsf{Verify}(vk, \mathbf{m}, \sigma) \to b \in \{0, 1\}$: output 1 if $\mathbf{m} \in \mathbb{Z}_p^n$, σ can be parsed as (W, y, z) and $e(W, UP_2^y) = e\left(P_1 H_0^z \prod_{i \geq 1} H_i^{m_i}, P_2\right)$, and otherwise 0.

Camenisch et al. [10, Lemma 1] proved that the BBS+ signature scheme is existentially unforgeable against chosen-message attacks under the qSDH assumption.

Proving Knowledge of BBS+ Signatures. Knowledge of a BBS+ Signature on a message tuple, i.e., of a witness for the language

$$(m_1, \ldots, m_n, \sigma = (W, y, z)) : e(W, UP_2^y) = e\left(P_1 H_0^z \prod_{i \geq 1} H_i^{m_i}, P_2\right)$$

can be proved in ZK (without disclosure of any m_i) as follows [10, Section 4.5].

The prover, in possession of a signature $\sigma = (W, y, z)$ on a message $\mathbf{m} \in \mathbb{Z}_p^n$, generates $r_1 \in_R \mathbb{Z}_p^*, r_2 \in_R \mathbb{Z}_p$, computes $W' = W^{r_1}$, $\overline{W} = W'^{-y} V^{r_1} (= W'^u)$, $V' = V^{r_1} H_0^{-r_2}$, sends those values to the verifier, sets $z' = z - r_2/r_1$, and thereafter performs a standard ZK Σ-protocol for the language

$$(m_1, \ldots, m_n, y, r_1, r_2, z') : W'^y \overline{W} H_0^{-r_2} = V',\ V'^{-1/r_1} H_0^{z'} H_1^{m_1} \cdots H_n^{m_n} = P_1^{-1}.$$

The verifier accepts the proof of knowledge of a signature if and only if the prior proof succeeds and $e(W', U) = e(\overline{W}, P_2)$. The signer is required to publish a pair (Q_1, Q_1^u) for some $Q_1 \neq 1_{\mathbb{G}_1}$. This pair is useful to ensure the ZK property of the proof.

2.4 Ciphertext-Policy Attribute-Based Encryption Schemes

Attribute-Based Encryption (ABE), introduced by Goyal et al. [16], enables to specify an access policy for encrypted data. In Ciphertext-Policy ABE (CP-ABE) schemes, the access policy is defined by an access structure embedded in the ciphertexts. One can decrypt a ciphertext generated by such a scheme only if one possesses a set of attributes that is accepted by the said access structure; in which case a secret key corresponding to those attributes can be requested to an authority in possession of a master secret key. Formally, a CP-ABE scheme is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ such that

$\mathsf{Setup}(1^\lambda, aux) \to (\mathcal{PP}, pk, msk)$: takes as an input a security parameter 1^λ and an auxiliary input aux (used to define attribute sets), and outputs public parameters, a public key and a master secret key;

$\mathsf{KeyDer}(msk, \mathcal{A}) \to sk_{\mathcal{A}}$: takes as an input a master secret key and a set of attributes, and outputs a corresponding secret key;

$\mathsf{Enc}(pk, m, \mathbb{S}) \to ct$: takes as an input a public key pk, a plaintext m and an access structure $\mathbb{S}$, and outputs a ciphertext ct; and

$\mathsf{Dec}(sk_{\mathcal{A}}, ct) \to m$: takes as an input a secret key corresponding to a set of a attributes $\mathcal{A}$ and a ciphertext, and outputs a plaintext m or $\perp$.

The CP-ABE schemes considered herein are required to be correct and adaptively Payload Hiding against Chosen-Plaintext Attacks [26, Definition 9] (or adaptively PH-CPA secure).

3 Definitions and Security Model

Leakage-deterring encryption (LDE) schemes are encryption schemes that deter users from sharing their decryption capabilities. To do so, user secrets are embedded into their public keys by an authority. As long as a user is honest, her secret remains private, but as soon as she produces a decryption device, anyone with access to it can recover her secret.

More precisely, we introduce Time-Based Leakage-Deterring Encryption (TB-LDE) schemes in which a third party $\mathcal{T}$ (which may as well be the embedding authority) assists users with decryption. The first time a user $\mathcal{U}$ wishes to decrypt a ciphertext in a given time period, she needs to request from $\mathcal{T}$ a key for the time period by executing a key-derivation protocol KeyDer. From then on until the end of the time period, *the decryption process is non-interactive.* The frequency at which the key-derivation protocol KeyDer is to be executed (i.e., the length of a time period, e.g., a week) can be adjusted at will by the system manager.

Yet, this does not affect in any way the ability to recover $\mathcal{U}$'s secret from a rogue decryption device B *at any time*: recovering a secret from an algorithm B will only be considered if it can decrypt ciphertexts for at least one time period subsequent to the current one, i.e., one for which $\mathcal{U}$ does not yet have a key. To recover a secret from a rogue algorithm B, one need not wait until

that future time period, it suffices to locally submit (i.e., without involving $\mathcal{T}$) to B ciphertexts encrypted for that future time period. This will prompt B to interact, and perform, with non-negligible probability, a valid zero-knowledge proof on secret of the user who owns B, allowing for the extraction of her secret.

3.1 Time-Based Leakage-Deterring Encryption Schemes

We now formally define TB-LDE schemes. Let $\mathcal{T} \subseteq \mathbb{N}$ denote a non-empty time-period set. Assume that all parties are roughly synchronized, i.e., that there is always a consensus among them on the current time period $t_c \in \mathcal{T}$ (to make sure that users cannot obtain keys for future time periods from the third-party). A TB-LDE scheme $\mathcal{E}$ consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, aux) \rightarrow (\mathcal{PP}, ck)$: an algorithm that generates public parameters and a commitment key on the input of a security parameter 1^λ and of an auxiliary input aux (used to define a time-period set $\mathcal{T}$ and other parameters)

$\mathsf{KeyGen.U}(\mathcal{PP}) \rightarrow (pk_{\mathcal{U}}, sk_{\mathcal{U}})$: a user key-generation algorithm

$\mathsf{KeyGen.T}(\mathcal{PP}) \rightarrow (pk_{\mathcal{T}}, sk_{\mathcal{T}})$: a user third-party key-generation algorithm

$\mathsf{KeyEn} = (\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}}), \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})) \rightarrow ((epk, esk), epk)$: a key-enhancement protocol between a user key-enhancement algorithm $\mathsf{KeyEn.U}$ and an authority key-enhancement algorithm $\mathsf{KeyEn.A}$. In addition to cryptographic keys, these algorithms take as an input a commitment c to a secret s, the secret s itself and an opening o. At the end of the protocol, $\mathsf{KeyEn.U}$ outputs a pair of "enhanced" keys (epk, esk), and $\mathsf{KeyEn.A}$ outputs epk

$\mathsf{Enc}(epk, pk_{\mathcal{T}}, m \in \mathcal{M}, t \in \mathcal{T}) \rightarrow ct$: a probabilistic encryption algorithm

$\mathsf{KeyDer} = (\mathsf{KeyDer.U}(esk, t), \mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck, t_c)) \rightarrow (sk_{\mathcal{T}}^{t,\mathcal{U}}, \perp)$: an interactive protocol between a user key-derivation algorithm $\mathsf{KeyDer.U}$ and a third-party key-derivation algorithm $\mathsf{KeyDer.T}$. *For every current time period t_c, if $t > t_c$, then $sk_{\mathcal{T}}^{t,\mathcal{U}} \leftarrow \perp$* (i.e., users cannot obtain keys for future time periods). Otherwise, at the end of the protocol, $\mathsf{KeyDer.U}$ outputs a third-party decryption key $sk_{\mathcal{T}}^{t,\mathcal{U}}$

$\mathsf{Dec}(esk, sk_{\mathcal{T}}^{t,\mathcal{U}}, ct) \rightarrow m$: a deterministic decryption algorithm

$\mathsf{Rec}(\mathsf{B}, epk, pk_{\mathcal{T}}, \mathcal{D}, t) \rightarrow s$: a recovery algorithm that takes as input an algorithm B (a "decryption box"), two keys epk and $pk_{\mathcal{T}}$, the description of a distribution $\mathcal{D}$ and a time period t, and outputs a secret s or $\perp$.

Commitment c may at first seem superfluous to the syntax, but the authority needs to receive some information bound to s (so that it can later be recovered given a decryption device) and that hides it (to ensure the privacy of s). Such information is nothing but a commitment.

3.2 Security Definitions

In this section, we define the security properties that an LDE scheme should satisfy. The security definitions are first given in a single-user case for simplicity, and are straightforwardly extended to the multi-user case in Appendix B.1. In every security experiment, the adversary is assumed to be stateful.

Correctness. Correctness states that the decryption of a plaintext encrypted for a certain time period, on the input of secret key for that time period, results in the plaintext with probability one. See Appendix B for a formal definition.

To model algorithms which can decrypt only certain ciphertexts, a "partial"-correctness definition must be given. In the following, for any two functions f and g of λ, the notation $f \gtrsim g$ means that there exists a negligible function negl such that $f \geq g - \text{negl}$.

Definition 1 (δ-Correctness). *For $\delta \in [0,1]$, given public keys epk and $pk_{\mathcal{T}}$, an algorithm B is said to be δ-correct in time period t with respect to a distribution $\mathcal{D}$ if for $m \leftarrow_{\$} \mathcal{D}$,*

$$\Pr\left[\mathsf{B}^{\mathsf{KeyDer.T}(sk_{\mathcal{T}},ck,t)}(\mathsf{Enc}(epk, pk_{\mathcal{T}}, m, t)) = m\right] \gtrsim \delta.$$

Note that the clock of algorithm KeyDer.T *is here set to time period t (so that it does not systematically reject every key request for future time periods).*

To define privacy and untraceability, consider the experiments in Figure 1.

$\mathbf{Exp}^{\text{priv}-\beta}_{\mathcal{E},\lambda}(\mathcal{A})$:	$\mathbf{Exp}^{\text{trace}-\beta}_{\mathcal{E},\lambda}(\mathcal{A})$:
1. $(\mathcal{PP}, ck) \leftarrow \mathsf{Setup}(1^\lambda)$ $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) \leftarrow \mathsf{KeyGen.U}(\mathcal{PP})$	1. $(\mathcal{PP}, ck) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(s_0, s_1) \leftarrow \mathcal{A}(ck, pk_{\mathcal{U}})$	2. $(esk_0, esk_1, t) \leftarrow \mathcal{A}(\mathcal{PP}, ck)$
3. $(c, o) \leftarrow \mathsf{Com}(ck, s_\beta)$	3. $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{U}}(esk_\beta, t)}$
4. $(\cdot, (epk, esk)) \leftarrow (\mathcal{A}(c), \mathsf{KeyEn.U}(ck, c, s_\beta, o, pk_{\mathcal{U}}, sk_{\mathcal{U}}))$	
5. $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{U}}(esk)}$	

Fig. 1. Privacy and traceability experiments for a TB-LDE scheme $\mathcal{E}$. In the privacy experiment, oracle $\mathcal{O}_{\mathcal{U}}(esk)$ can be requested to execute either $\mathsf{KeyDer.U}(esk, \cdot)$ on arbitrary time periods or $\mathsf{Dec}(esk, \cdot, \cdot)$ on arbitrary third-party derived keys and ciphertexts, and return the outputs to $\mathcal{A}$. In the traceability experiment, oracle $\mathcal{O}_{\mathcal{U}}(esk_\beta, t)$ runs $\mathsf{KeyDer.U}(esk_\beta, t)$.

Privacy. Privacy guarantees that not even the authority that the user interacts with in the key-enhancement protocol can infer any information about her secret.

Definition 2 (Privacy (of the User Secret)). *$\mathcal{E}$ satisfies privacy of user secrets if for every efficient adversary $\mathcal{A}(1^\lambda)$, there exists a negligible function* negl *such that*

$$\mathbf{Adv}^{\text{priv}}_{\mathcal{E},\lambda}(\mathcal{A}) = \left|\Pr\left[\mathbf{Exp}^{\text{priv}-0}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{priv}-1}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right]\right| \leq \text{negl}(\lambda).$$

LD-IND-CCA Security. As for classical cryptosystems, the secrecy of the user's messages should be retained even when the key-enhancement protocol is taken into account. This requirement is captured by the LD-IND-CCA property.

Definition 3 (LD-IND-CCA Security). *$\mathcal{E}$ satisfies Leakage-Deterring Indistinguishability under Chosen-Ciphertext Attacks (LD-IND-CCA) if for every efficient adversary $\mathcal{A}(1^\lambda)$, there exists a negligible function* negl *such that*

$$\mathbf{Adv}^{\mathrm{ld-cca}}_{\mathcal{E},\lambda}(\mathcal{A}) = \left| \Pr\left[\begin{array}{c} (\mathcal{PP}, ck) \leftarrow \mathsf{Setup}(1^\lambda),\ (pk_{\mathcal{U}}, sk_{\mathcal{U}}) \leftarrow \mathsf{KeyGen.U}(\mathcal{PP}), \\ (c, s, o) \leftarrow \mathcal{A}(ck, pk_{\mathcal{U}}), \\ b' = b\colon\ (\cdot, (epk, esk)) \leftarrow (\mathcal{A}(c),\ \mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}})), \\ (m_0, m_1, t, pk_{\mathcal{T}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{U}_1}(esk)}(epk),\ b \in_R \{0,1\}, \\ ct^* \leftarrow \mathsf{Enc}(epk, pk_{\mathcal{T}}, m_b, t),\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{U}_2}(esk, ct^*)}(ct^*) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda),$$

with $\mathcal{O}_{\mathcal{U}_1}(esk)$ an oracle that can be requested by $\mathcal{A}$ to execute either $\mathsf{KeyDer.U}(esk, \cdot)$ *or* $\mathsf{Dec}(esk, \cdot, \cdot)$ *and return the outputs to $\mathcal{A}$, and $\mathcal{O}_{\mathcal{U}_2}(esk, ct^*)$ an oracle that can be requested by $\mathcal{A}$ to execute either* $\mathsf{KeyDer.U}(esk, \cdot)$ *or* $\mathsf{Dec}(esk, \cdot, \cdot)$ *on arbitrary ciphertexts ct such that $ct_0 \neq ct_0^*$ (only the first part of the ciphertext, the one that the user can decrypt on her own, must be different) and return the outputs.*

Untraceability. Untraceability ensures that the protocol in which the third party helps the user decrypt ciphertexts should preserve her anonymity.

Definition 4 (Untraceability). *$\mathcal{E}$ satisfies untraceability if for every efficient adversary $\mathcal{A}(1^\lambda)$, there exists a negligible function* negl *such that*

$$\mathbf{Adv}^{\mathrm{trace}}_{\mathcal{E},\lambda}(\mathcal{A}) = \left| \Pr\left[\mathbf{Exp}^{\mathrm{trace}-0}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{trace}-1}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right] \right| \leq \mathrm{negl}(\lambda).$$

Recoverability. Given that the authority correctly executes its key-enhancement algorithm and that the third party keys are correctly generated, a TB-LDE scheme should ensure that the secret output by the recovery algorithm is the one the user committed to during the key-enhancement protocol. Note that the truthful generation of third party keys is necessary to enforce that the user, to decrypt ciphertexts, must to interact at least once per time period.

Definition 5 (Recoverability (of the User Secret)). *$\mathcal{E}$ satisfies (rewinding black-box) recoverability of the user secret (with respect to a distribution class $\mathscr{D}$) if for every efficient adversary $\mathcal{A}(1^\lambda)$, for every current time period t_c, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} s \neq s', \mathcal{D} \in \mathscr{D}, t > t_c, \mathsf{B} \textit{ is } \delta\textit{-correct in time period } t \textit{ w.r.t. } \mathcal{D}\colon \\ \quad (\mathcal{PP}, ck) \leftarrow \mathsf{Setup}(1^\lambda),\ (pk_{\mathcal{T}}, sk_{\mathcal{T}}) \leftarrow \mathsf{KeyGen.T}(\mathcal{PP}), \\ \quad (s, pk_{\mathcal{U}}) \leftarrow \mathcal{A}(\mathcal{PP}, ck, pk_{\mathcal{T}}),\ (c, o) \leftarrow \mathsf{Com}(ck, s), \\ \quad (\cdot, epk) \leftarrow (\mathcal{A}(c, o),\ \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})), \\ \quad (\mathsf{B}, \mathcal{D}, t) \leftarrow \mathcal{A}^{\mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck, t_c)},\ s' \leftarrow \mathsf{Rec}(\mathsf{B}, epk, pk_{\mathcal{T}}, \mathcal{D}, t) \end{array} \right] \leq \mathrm{negl}(\lambda).$$

Remark 1. The maximal class $\mathscr{D}$ for which it is possible to successfully deter users from delegating their decryption capabilities is the class of distributions

$\mathcal{D}$ such that $\delta > p(\mathcal{D})$ and $\delta - p(\mathcal{D})$ is non-negligible. Indeed, for a distribution $\mathcal{D}$ such that $p(\mathcal{D}) > \delta$ or $\delta - p(\mathcal{D})$ is negligible, any decryption algorithm B can merely output a message with probability mass $p(\mathcal{D})$, and readily satisfy δ-correctness as pointed out by Kiayias and Tang [18]; in which case recoverability cannot be achieved since the user secret is not involved in the decryption process.

Remark 2. The restriction $t > t_c$ is crucial. Would it not be the case, an adversary could request a decryption key for a time period $t \leq t_c$ from third-party $\mathcal{T}$, hard-code it in B, and thereby achieve 1-correctness for time period t without ever having to interact.

4 Generic Construction of a TB-LDE Scheme

In this section, we give a generic construction which turns any PKE scheme into a TB-LDE scheme. The main ideas are as follows. We use the original PKE scheme to encrypt a one-time-pad encryption of the plaintext, and encrypt the one-time-pad key with a CP-ABE scheme for which third party $\mathcal{T}$ holds the master secret key. The CP-ABE allows to specify a recipient user and a time-period in which the whole ciphertext can be decrypted. Users are thereby compelled to interact with the third party at least once per time-period to obtain the secret key corresponding to the time period indicated by the ciphertext policy. This interaction requires users to perform a proof on the secrets to which they committed during the key-enhancement protocol, and it allows, via rewinding, for the recoverability of those secrets by anyone with a decryption device.

In more detail, during the key-enhancement protocol, a user $\mathcal{U}$ commits to a secret, sends a commitment to the authority, and after proving knowledge of the secret to the latter, she receives a random identity and a signature on it and the commitment. Ciphertexts consist of two parts: a first that $\mathcal{U}$ can decrypt on her own and another (the CP-ABE part) that she can only decrypt with a key derived from $\mathcal{T}$'s master secret key. To obtain such a key, $\mathcal{U}$ encrypts her identity and must prove, to $\mathcal{T}$, knowledge of a signature on the encrypted identity and on a commitment to which she knows an opening. As the signature scheme is assumed unforgeable, $\mathcal{U}$ has to use the commitment that she sent to the authority during the key-enhancement protocol. Any pirate decryption algorithm B that can decrypt a non-negligible amount of ciphertexts generated with $\mathcal{U}$'s key must perform the same proof with non-negligible probability. The extractor of the proof system can then recover $\mathcal{U}$'s secret.

Formal Description. Let $\mathcal{T}$ be a set of natural integers, the time-period set, and $\mathcal{ID}$ a non-empty identity set. Our construction uses as building blocks

- $\mathcal{C}$ a commitment scheme with which users commit to their secrets,
- $\mathcal{S}$ a signature scheme used to sign user commitments and identities
- $\mathcal{E}_0$ a public-key encryption scheme to compute ciphertext parts that the user can decrypt on her own,

– $\mathcal{E}_1$ a CP-ABE scheme with attribute space $\mathcal{T} \times \mathcal{ID}$ and equality as access policy used to compute the ciphertext parts for which $\mathcal{U}$ needs assistance from $\mathcal{T}$,
– $\mathcal{E}_2$ a PKE scheme with message space $\mathcal{ID}$ is used to encrypt $\mathcal{U}$'s identity when she interacts with $\mathcal{T}$.

Suppose that $\mathcal{E}_0$ and $\mathcal{E}_1$ share the same message space $\mathcal{M}$, on which there exists an internal composition law $\oplus$ such that for all $m \in \mathcal{M}$, the map $\cdot \oplus m$ is a permutation of $\mathcal{M}$. Let $\cdot \ominus m$ stand for its inverse. To turn the key-derivation algorithm of CP-ABE $\mathcal{E}_1$ into an interactive protocol between $\mathcal{U}$ and $\mathcal{T}$, assume that there exists probabilistic algorithms Der_0 and Der_2 (Der_2 will be used for simulation in the proof of recoverability – see Theorem 5), and a deterministic algorithm Der_1 such that

1. for all $sk_{\mathcal{T}}$, t, $(ek, dk) \leftarrow \mathcal{E}_2.\mathsf{KeyGen}(\mathcal{PP})$, id, $ct^{id} \leftarrow \mathcal{E}_2.\mathsf{Enc}(ek, id; r_{\mathcal{U}})$, variables $(ek, dk, r_{\mathcal{U}}, \mathsf{Der}_0(sk_{\mathcal{T}}, t, ek, ct^{id}))$ and $(ek, dk, r_{\mathcal{U}}, \mathsf{Der}_2(sk_{\mathcal{T}}^{t,id}, ek))$ have the same distribution; and
2. for all $sk_{\mathcal{T}}$, t, $(ek, dk) \leftarrow \mathcal{E}_2.\mathsf{KeyGen}(\mathcal{PP})$, id, $ct^{id} \leftarrow \mathcal{E}_2.\mathsf{Enc}(ek, id)$, $sk_{\mathcal{T}}'^{t,id} \leftarrow \mathsf{Der}_0(sk_{\mathcal{T}}, t, ek, ct^{id})$, $\mathsf{Der}_1(dk, sk_{\mathcal{T}}'^{t,id}) = \mathcal{E}_1.\mathsf{KeyDer}(sk_{\mathcal{T}}, \{t, id\})$.

We then construct a TB-LDE scheme $\mathcal{E}$, parametrized by the time-period set $\mathcal{T}$ and the identity set $\mathcal{ID}$, such that

$\mathsf{Setup}(1^\lambda, (\mathcal{T}, \mathcal{ID})) \to (\mathcal{PP}, ck)$: generates public parameters, by running algorithms $\mathcal{E}_0.\mathsf{Setup}(1^\lambda)$, $\mathcal{E}_1.\mathsf{Setup}(1^\lambda, (\mathcal{T}, \mathcal{ID}))$, $\mathcal{C}.\mathsf{Setup}(1^\lambda)$, $\mathcal{S}.\mathsf{Setup}(1^\lambda)$, and computes a commitment key $ck \leftarrow \mathsf{ComKeyGen}(\mathcal{PP})$

$\mathsf{KeyGen.U}(\mathcal{PP}) \to (pk_{\mathcal{U}}, sk_{\mathcal{U}})$: runs $\mathcal{E}_0.\mathsf{KeyGen}(\mathcal{PP})$

$\mathsf{KeyGen.T}(\mathcal{PP}) \to (pk_{\mathcal{T}}, sk_{\mathcal{T}})$: runs $(pk_{\mathcal{T}}, sk_{\mathcal{T}}) \leftarrow (pk, msk) \leftarrow \mathcal{E}_1.\mathsf{KeyGen}(\mathcal{PP})$

$\mathsf{KeyEn} = (\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}}), \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})) \to ((epk, esk), epk)$: is the following protocol between $\mathsf{KeyEn.U}$ and $\mathsf{KeyEn.A}$:

1. $\mathsf{KeyEn.U}$ and $\mathsf{KeyEn.A}$ run protocol $\mathsf{PoK}\{(s, o) \colon \mathsf{Open}(ck, c, s, o) = 1\}$ as prover and verifier respectively. If it fails, the overall key-enhancement protocol is aborted, i.e., $epk \leftarrow esk \leftarrow \perp$, otherwise
2. $\mathsf{KeyEn.A}$ generates $(sk, vk) \leftarrow \mathcal{S}.\mathsf{KeyGen}(\mathcal{PP})$, $id \in_R \mathcal{ID}$, computes $\sigma = \mathsf{Sign}(sk, (c, id))$, sets $epk \leftarrow (pk_{\mathcal{U}}, c, id, \sigma)$, and sends id and σ to $\mathsf{KeyEn.U}$
3. $\mathsf{KeyEn.U}$ sets epk as $\mathsf{KeyEn.A}$, and sets $esk \leftarrow (sk_{\mathcal{U}}, c, s, o, id, \sigma)$.

$\mathsf{Enc}(epk, pk_{\mathcal{T}}, m \in \mathcal{M}, t \in \mathcal{T}) \to ct$: generates $m_1 \in_R \mathcal{M}$, sets $m_0 \leftarrow m \oplus m_1$, and outputs $ct \leftarrow (\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, m_0), \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, m_1, \mathbb{S} = \{t, id\}))$

$\mathsf{KeyDer} = (\mathsf{KeyDer.U}(esk, t), \mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck, vk, t_c)) \to (sk_{\mathcal{T}}^{t,id}, \perp)$: is a two-party interactive equivalent of algorithm $\mathcal{E}_1.\mathsf{KeyDer}(sk_{\mathcal{T}}, \{t, id\})$. More precisely, it is a protocol in which $\mathsf{KeyDer.U}$ and $\mathsf{KeyDer.T}$ proceed as follows.

1. $\mathsf{KeyDer.U}$ generates and stores a pair of keys $(ek, dk) \leftarrow \mathcal{E}_2.\mathsf{KeyGen}(\mathcal{PP})$ if none was priorly stored (otherwise reuses such a pair), computes $ct^{id} \leftarrow \mathcal{E}_2.\mathsf{Enc}(ek, id; r_{\mathcal{U}})$, and sends (ek, t, ct^{id}) to $\mathsf{KeyDer.T}$

2. if $t > t_c$, then KeyDer.T returns $\perp$, and KeyDer.U outputs $\perp$; otherwise algorithms KeyDer.U and KeyDer.T run protocol

$$\mathsf{PoK}\{(c, s, o, id, \sigma, r_{\mathcal{U}}) : \mathsf{Open}(ck, c, s, o) = 1, \mathsf{Verify}(vk, (c, id), \sigma) = 1, ct^{id} = \mathcal{E}_2.\mathsf{Enc}(ek, id; r_{\mathcal{U}})\} \quad (1)$$

as prover and verifier respectively. If protocol PoK fails, the overall protocol is aborted, i.e., $sk_{\mathcal{T}}^{t,id} \leftarrow \perp$; otherwise

3. KeyDer.T computes $sk_{\mathcal{T}}'^{t,id} \leftarrow \mathsf{Der}_0(sk_{\mathcal{T}}, t, ek, ct^{id})$ and sends it to KeyDer.U
4. KeyDer.U outputs $sk_{\mathcal{T}}^{t,id} = \mathsf{Der}_1(dk, sk_{\mathcal{T}}'^{t,id})$.

$\mathsf{Dec}(esk, sk_{\mathcal{T}}^{t,id}, ct) \rightarrow m$: returns $m = \mathcal{E}_0.\mathsf{Dec}(sk, ct_0) \ominus \mathcal{E}_1.\mathsf{Dec}(sk_{\mathcal{T}}^{t,id}, ct_1)$ (outputs $\perp$ instead if either $\mathcal{E}_0.\mathsf{Dec}(sk, ct_0) = \perp$ or $\mathcal{E}_1.\mathsf{Dec}(sk_{\mathcal{T}}^{t,id}, ct_1) = \perp$)

$\mathsf{Rec}(\mathsf{B}, epk, pk_{\mathcal{T}}, \mathcal{D}, t) \rightarrow s$: generates messages $m \leftarrow_{\$} \mathcal{D}$, computes $\mathsf{Enc}(epk, pk_{\mathcal{T}}, m, t)$ and runs B on it until the latter engages in protocol PoK as prover, and succeeds in it. Once this event occurs (it is yet to be proved that it does indeed occur), algorithm Rec runs extractor $\mathcal{K}$, which can rewind B, to extract a witness that contains a secret s to which c is a commitment. Note that algorithm Rec runs B *locally*, i.e., B does not interact with KeyDer.T, but rather Rec which plays the role of the verifier. *Importantly,* Rec *does not reject the key-request from* B *if it is for a time period* $t > t_c$. This allows for recoverability without having to wait until a future time period in which B is claimed to be δ-correct.

Correctness & Security. We now state the security properties achieved by our construction.

Theorem 1 (Correctness). *$\mathcal{E}$ is correct if $\mathcal{E}_0$ and $\mathcal{E}_1$ are correct and if* PoK *is complete.*

Proof. If PoK is complete, then, for any ciphertext, KeyDer.U successfully obtains a secret key corresponding to the time period indicated in the access structure of the ciphertext. The correctness of $\mathcal{E}_0$ and $\mathcal{E}_1$ then implies that of $\mathcal{E}$. □

Complete proofs for Theorems 2, 3 and 4 are given in Appendix. C

Theorem 2 (Privacy). *$\mathcal{E}$ satisfies privacy if $\mathcal{C}$ is hiding, protocol* PoK *is zero knowledge, and $\mathcal{E}_2$ is IND-CPA secure.*

Proof (Sketch). If the commitment scheme is hiding, the authority cannot infer any information about user secrets. If PoK is zero-knowledge and $\mathcal{E}_2$ is IND-CPA, the third-party cannot infer any information about the user identities related which are related to commitments to user secrets. □

Theorem 3 (LD-IND-CCA Security). *$\mathcal{E}$ is LD-IND-CCA secure if $\mathcal{E}_0$ is IND-CCA secure.*

Proof (Sketch). Reducing the the LD-IND-CCA security of $\mathcal{E}$ to the IND-CCA security of $\mathcal{E}_0$ is straightforward since without the user public key, even the third party can tell apart an encryption of $m \oplus m_0$ from an encryption of $m \oplus m_1$, where m is a uniformly random bit-string, only with non-negligible probability.

Theorem 4 (Untraceability). *$\mathcal{E}$ satisfies untraceability if proof system* PoK *is zero knowledge and $\mathcal{E}_2$ is IND-CPA secure.*

Proof (Sketch). As users encrypt their identities to request keys, if $\mathcal{E}_2$ is IND-CPA secure and PoK is ZK, then the third-party cannot infer any information about the user identities during protocol KeyDer. □

Theorem 5 (Recoverability). *$\mathcal{E}$ satisfies recoverability with respect to the class of distributions $\mathcal{D}$ such that $\delta > p(\mathcal{D})$ and $\delta - p(\mathcal{D})$ is non-negligible assuming $\mathcal{C}$ to be binding, $\mathcal{S}$ to be existentially unforgeable and $\mathcal{E}_1$ to be adaptively PH-CPA secure.*

Proof (Sketch). It suffices to prove that with a probability close to δ, when given ciphertexts generated for the time period and the identity for which Rec is δ-correct, and encrypting messages with distribution $\mathcal{D}$, algorithm B requests the third-party secret key corresponding to the time period and the identity. This is the crucial part of the proof.

As soon as this event occurs, algorithm Rec runs extractor $\mathcal{K}$ to extract a secret. Since the commitment and the identity used in the witness for the proof are signed by the key-enhancement authority, algorithm Rec must send, with overwhelming probability, the commitment and an encryption of the identity that are in the user enhanced public key. As the commitment scheme is binding, the extracted secret is the one that was used in the key-enhancement protocol. □

5 Instantiation

We now instantiate each of the building blocks of the construction in Section-4.

Let $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e)$ be a pairing group such that the DLOG assumption holds in $\mathbb{G}_1$, the Decisional Diffie–Hellman (DDH) assumption holds in $\mathbb{G}_2$ and $\mathbb{G}_T$, the DLIN assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$, and the qSDH problem in intractable in $(\mathbb{G}_1, \mathbb{G}_2)$ (see Appendix 2.2). Set

- $\mathcal{C}$ as the standard Pedersen commitment scheme over $\mathbb{G}_1$
- $\mathcal{E}_0$ as the Cramer–Shoup encryption scheme (Appendix A.1) with message space $\mathbb{G}_T$
- $\mathcal{E}_1$ as the Okamoto–Takashima CP-ABE scheme (Appendix A.3)
- $\mathcal{E}_2$ as the Elgamal encryption scheme with message space $\mathbb{Z}_p$ and $\mathcal{S}$ as the BBS+ signature scheme (Section 2.3) in the case $n = 3$ (to sign a secret s, an opening o and an identity id).

Note that the Okamoto–Takashima scheme remains PH-CPA secure under the same assumptions if generator $\mathcal{G}_{\mathrm{ob}}$ outputs matrices $\mathbf{V}_k$ instead of matrices $\mathbf{B}_k^*$, and its setup algorithm includes matrices $\hat{\mathbf{V}}_k$ (defined similarly to $\hat{\mathbf{B}}_k^*$) instead of matrices $\hat{\mathbf{B}}_k^*$ in the master secret key. In this section, the Okamato–Takashima scheme is considered with this modification.

It remains to provide algorithms Der_0, Der_1, Der_2 for protocol KeyDer and a proof system (P, V) for the language of Equation-1. Consider then the following algorithms:

$\mathsf{Der}_0(sk_{\mathcal{T}}, t, ek = (P_2, Q_2), ct^{id} = (C_0^{id}, C_1^{id})) \to sk_{\mathcal{T}}'^{t,id}$: generate $\lambda_1, \ldots, \lambda_7, \alpha, y_0 \in \mathbb{Z}_p$, $\mathbf{y}_1, \mathbf{y}_2 \in_R \mathbb{Z}_p^2$, compute $\mathbf{k}_0^* = \left[\alpha\ 1\ y_0\right] \hat{\mathbf{B}}_0^*$, $\mathbf{k}_1^* = \left[\alpha \mathbf{x}_1\ \mathbf{y}_1\right] \hat{\mathbf{B}}_1^*$,

$$\mathbf{k}_2'^* = \left[\left(P_2^{\lambda_j} \left(C_0^{id}\right)^{\alpha \hat{\mathbf{V}}_{2,2,j}}, Q_2^{\lambda_j} P_2^{\alpha \hat{\mathbf{V}}_{2,1,j}} \left(C_1^{id}\right)^{\alpha \hat{\mathbf{V}}_{2,2,j}} P_2^{\mathbf{y}_{2,1} \hat{\mathbf{V}}_{2,3,j}} P_2^{\mathbf{y}_{2,2} \hat{\mathbf{V}}_{2,4,j}}\right)\right]_j,$$

and return $sk_{\mathcal{T}}'^{t,id} = (\mathbf{k}_0^*, \mathbf{k}_1^*, \mathbf{k}_2'^*)$

$\mathsf{Der}_1(dk = q = \mathrm{dlog}_{P_2}(Q_2), sk_{\mathcal{T}}'^{t,id}) \to sk_{\mathcal{T}}^{t,id}$: parse $\mathbf{k}_2'^*$ as $\left[\left(\mathbf{k}_{2,j,0}'^*, \mathbf{k}_{2,j,1}'^*\right)\right]_j$, compute $\mathbf{k}_2^* = \left[\mathbf{k}_{2,j,1}'^* / \left(\mathbf{k}_{2,j,0}'^*\right)^q\right]_j$, and return $sk_{\mathcal{T}}^{t,id} = (\mathbf{k}_0^*, \mathbf{k}_1^*, \mathbf{k}_2^*)$

$\mathsf{Der}_2(sk_{\mathcal{T}}^{t,id}, ek) \to sk_{\mathcal{T}}'^{t,id}$: generate $\lambda_1, \ldots, \lambda_7 \in_R \mathbb{Z}_p$, compute a vector $\mathbf{k}_2'^* = \left[\left(P_2^{\lambda_j}, Q_2^{\lambda_j} \mathbf{k}_{2,j}^*\right)\right]_j$, and return $sk_{\mathcal{T}}'^{t,id} = (\mathbf{k}_0^*, \mathbf{k}_1^*, \mathbf{k}_2'^*)$.

Notice that requirements in Section 4 for Der_0, Der_1 and Der_2 are met.

Moreover, in the key-derivation protocol KeyDer, protocol PoK should be a ZKPoK protocol for the language

$$(s, o, \overbrace{\mathbf{x}_{2,2}}^{id}, \overbrace{W, y, z}^{\sigma}, \overbrace{\gamma}^{r_{\mathcal{U}}}) \colon\ e(W, UP_2^y) = e(P_1 H_0^z H_1^r H_2^s H_3^{\mathbf{x}_{2,2}}, P_2),\ C_0^{id} = P_2^{\gamma},\ C_1^{id} = Q_2^{\gamma} P_2^{\mathbf{x}_{2,2}}.$$

A standard ZK protocol for proving knowledge of a preimage of a group homomorphism is then a suitable ZKPoK protocol.

Theorems 1, 2, 3, 4, 5 imply that this instantiation satisfies correctness, privacy, LD-IND-CCA security, untraceability and recoverability.

	$\mathbb{Z}$	$\mathbb{Z}_p$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$
Keys and Ciphertexts					
$pk_{\mathcal{U}}$	1	8	2	0	5
$sk_{\mathcal{U}}$	1	10	1	0	0
$pk_{\mathcal{T}}$	3	0	57	0	1
$sk_{\mathcal{T}}$	3	71	0	0	0
$sk_{\mathcal{T}}^{t,id}$	0	0	0	19	0
ct	0	2	19	0	5
Protocols					
KeyEn	0	6	1	0	0
KeyDer	0	12	2	32	0

	(Multi-)Exp.	Pairing
Der_0	26 in $\mathbb{G}_2$	0
Der_1	7 in $\mathbb{G}_2$	0
Enc	27 (26 in $\mathbb{G}_1$, 1 in $\mathbb{G}_T$)	0
Dec	1 (in $\mathbb{G}_T$)	19

Fig. 2. Efficiency of our instantiation. On the left, for keys and ciphertexts, the figures represent the number of elements they comprise. As for the protocols, it is their bandwidth that is represented, i.e., the number of elements exchanged between the parties. The table on the right indicates the number of operations (multi-exponentiation and pairing) performed by each of the algorithms.

Comparison with Kiayias and Tang's Generic Construction. The only existing leakage-deterring scheme in the literature is Kiayias and Tang's. They did not provide an instantiation of their generic construction of an LD-IND-CPA scheme [18, Section 4]. Yet, it can naturally be instantiated with the Elgamal encryption scheme. In that case, for secrets of 128 bits, a partial correctness $\delta = 4/2^3 = 50\%$ and the error correcting code that they propose [17, Figure 1, No 1], their enhanced public keys consist of $162 = 2*81$ Elgamal public keys and ciphertexts consist of $30 = 2 * 15$ Elgamal ciphertexts (i.e., 60 group elements). To make their scheme LD-IND-CCA secure, they compose it with a standard CCA-secure scheme which must then encrypt a vector of group elements, and the keys of which must be accounted for in the overall enhanced public key.

Section 6 gives construction a more efficient construction of our scheme in the ROM. The message space of $\mathcal{E}_0$ is $\mathbb{G}_1$ and the ciphertexts consist of 2 $\mathbb{Z}_p$ elements, 23 $\mathbb{G}_1$ and 1 $\mathbb{G}_T$ elements. Our enhanced public keys and ciphertexts in the ROM are therefore much shorter than those of Kiayias and Tang. However, computing our ciphertexts is more expensive because of the pairings of the Okamoto–Takashima CP-ABE scheme.

6 Construction in the Random Oracle Model

The main drawback, in terms of computation time and ciphertext size, of the Section-4 construction is that the message spaces of $\mathcal{E}_0$ and $\mathcal{E}_1$ must match. As a result, the Section-5 instantiation requires the Cramer–Shoup encryption scheme (with ciphertexts consisting of 4 plaintext-group elements) to have $\mathbb{G}_T$ as message space since the message space of the Okamoto–Takashima CP-ABE is $\mathbb{G}_T$. However, target group elements are typically large (around 10 times larger than $\mathbb{G}_1$ elements) and operations in $\mathbb{G}_T$ are much slower than in $\mathbb{G}_1$. It would

thus be preferable to have $\mathbb{G}_1$ as a message space for $\mathcal{E}_0$. Consider then the following construction which is proved secure in the ROM. The RO is further denoted by Hash). It is identical to that of Section 4 except for its message space that is now $\{0,1\}^{n(\lambda)}$, the message space $\mathcal{M}_0$ of $\mathcal{E}_0$, the message space $\mathcal{M}_1$ of $\mathcal{E}_1$, and for algorithms Enc and Dec. Those latter algorithms are now[6]

$\mathsf{Enc}(epk, pk_{\mathcal{T}}, m \in \{0,1\}^n, t \in \mathcal{T}) \rightarrow ct:$ that generates $K_{\mathcal{U}} \in_R \mathcal{M}_0$, $K_{\mathcal{T}} \in_R \mathcal{M}_1$, and outputs a ciphertext $ct = (\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, K_{\mathcal{U}}), \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, K_{\mathcal{T}}, \{t, id\}), m \oplus \mathsf{Hash}(K_{\mathcal{U}}, K_{\mathcal{T}}))$;

$\mathsf{Dec}(esk, ct) \rightarrow m:$ an algorithm that parses ct as (ct_0, ct_1, ct_2), and computes and outputs $ct_2 \oplus \mathsf{Hash}(\mathcal{E}_0.\mathsf{Dec}(sk_{\mathcal{U}}, ct_0), \mathcal{E}_1.\mathsf{Dec}(sk_{\mathcal{T}}^{t,id}, ct_1))$.

With this alteration, the proofs of correctness, privacy and untraceability are straightforwardly adapted from those of the scheme in Section 4. As for LD-IND-CCA security and recoverability, their proofs require a more elaborate (although simple) argumentation, and are given below. Note that still no proof of knowledge on the ciphertexts is required (only on an encrypted identity at the beginning of each time period). Therefore, no proof of circuit satisfiability for the RO must be computed.

Theorem 6. *$\mathcal{E}$ is LD-IND-CCA secure in the PROM if $\mathcal{E}_0$ is IND-CCA secure.*

Proof. The IND-CCA security of $\mathcal{E}$ can be reduced to the LD-IND-CCA security of $\mathcal{E}_0$. Indeed, if $\mathcal{A}$ is an adversary for the LD-IND-CCA security game, consider $\mathcal{S}$ an algorithm which interacts with $\mathcal{A}$ and the challenger $\mathcal{C}$ of the IND-CCA security game. Algorithm $\mathcal{S}$ proceeds like the simulator of theorem 3 during the key-generation phase and the key-enhancement protocol. In the first query phase, whenever $\mathcal{A}$ asks for the decryption of a ciphertext $ct = (ct_0, ct_1, ct_2)$, algorithm $\mathcal{S}$ requests the decryption of ct_0 to $\mathcal{C}$, and can answer the query of $\mathcal{A}$. Upon receiving a challenge tuple $(m_0, m_1, t, pk_{\mathcal{T}})$ from $\mathcal{A}$, algorithm $\mathcal{S}$ generates generates $K_{\mathcal{U},0}, K_{\mathcal{U},1} \in_R M_0$, $K_{\mathcal{T}} \in_R \mathcal{M}_1$, and if $\mathsf{Hash}(K_{\mathcal{U},b}, K_{\mathcal{T}})$ for $b \in \{0,1\}$ has been queried in the first query phase, it generates new random values. This does not blow up the running time of $\mathcal{S}$ since $\mathcal{A}$ is efficient and cannot have made an amount of RO queries that is non-negligible compared to $|\mathcal{M}_0||\mathcal{M}_1|$. Algorithm $\mathcal{S}$ also generates $R \in_R \{0,1\}^n$, sends $(K_{\mathcal{U},0}, K_{\mathcal{U},1})$ to $\mathcal{C}$ as a challenge pair, receives a challenge ciphertext ct_0^*, computes $ct_1^* = \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, K_{\mathcal{T}}, \{t, id\})$, and returns (ct_0^*, ct_1^*, R) to $\mathcal{A}$. In the second query phase, $\mathcal{S}$ proceeds as in the first, except that if $\mathcal{A}$ queries $\mathsf{Hash}(K_{\mathcal{U},b}, K_{\mathcal{T}})$ to the RO for $b \in \{0,1\}$, algorithm $\mathcal{S}$ programs $\mathsf{Hash}(K_{\mathcal{U},b}, K_{\mathcal{T}}) = m_b \oplus R$. Besides, as $\mathcal{A}$ cannot request the decryption of a ciphertext ct such that $ct_0 = ct_0^*$ by definition, algorithm $\mathcal{S}$ can always answer decryption queries in this second phase. Algorithm $\mathcal{S}$ ultimately forwards the guess of $\mathcal{A}$ to $\mathcal{C}$. As $\mathcal{S}$ perfectly simulates the LD-IND-CCA-game challenger to $\mathcal{A}$, its advantage in the IND-CCA game is at least that of $\mathcal{A}$ in the LD-IND-CCA game. If the latter were non-negligible, then so would be former, and the IND-CCA security of $\mathcal{E}_0$ would be contradicted. □

[6] $\oplus$ here denotes the traditional XOR operation.

Theorem 7. *$\mathcal{E}$ satisfies recoverability in the PROM with respect to the class of distributions $\mathcal{D}$ such that $\delta > p(\mathcal{D})$ and $\delta - p(\mathcal{D})$ is non-negligible assuming $\mathcal{C}$ to be binding, $\mathcal{S}$ to be existentially unforgeable and $\mathcal{E}_1$ to be adaptively PH-CPA secure.*

Proof. The proof is conducted in the very same fashion as in the proof of Theorem 5, except for the hybrids: $\mathcal{H}_0$ submits regular ciphertexts to B and $\mathcal{H}_1$ submits $(\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, K_{\mathcal{U}}), \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, K'_{\mathcal{T}}, \{t, id\}), m \oplus \mathsf{Hash}(K_{\mathcal{U}}, K_{\mathcal{T}}))$ for a uniformly random $K'_{\mathcal{T}}$. The inequality

$$\left|\Pr\left[\mathsf{B}(\mathcal{H}_0) = m|\overline{F}\right] - \Pr\left[\mathsf{B}(\mathcal{H}_1) = m|\overline{F}\right]\right| \leq \mathbf{Adv}^{\mathrm{ph-cpa}}_{\mathcal{E}_1,\lambda}(\mathcal{S})$$

is proved in the same manner, and $\Pr\left[\mathsf{B}(\mathcal{H}_1) = m|\overline{F}\right] \leq p(\mathcal{D}) + 1/|\mathcal{M}_{\mathcal{T}}|$ as B can only either guess m with probability at most $p(\mathcal{D})$ or guess $K_{\mathcal{T}}$ with probability at most $1/|\mathcal{M}_{\mathcal{T}}|$. As $1/|\mathcal{M}_{\mathcal{T}}|$ is negligible in λ, the rest of the proof remains the same. □

7 Non–Time-Based LDE Schemes

In case only few ciphertexts are expected to be decrypted, communication during decryption might not be a major hindrance. LDE schemes in which users need to interact whenever they wish to decrypt a ciphertext are conceivable, and could spare the need for a CP-ABE scheme which incurs larger keys and ciphertexts than regular encryption schemes. Time periods and synchronization would then be unnecessary. In such a system, compared to the Section-3 definition of a TB-LDE scheme, $\mathcal{T}$ does not send a key to $\mathcal{U}$ (i.e., protocol KeyDer is obsolete), and $\mathsf{Dec} = (\mathsf{Dec.U}(esk, ct), \mathsf{Dec.T}(sk_{\mathcal{T}}, ck))$ is an interactive protocol between a user decryption algorithm $\mathsf{Dec.U}$ and a third-party decryption algorithm $\mathsf{Dec.T}$, at the end of which $\mathsf{Dec.U}$ outputs a plaintext m or $\perp$, and $\mathsf{Dec.T}$ outputs $\perp$.

7.1 Security Definitions

The privacy definition remains the same, the traceability experiments and the LD-IND-CCA security game are only modified not to incorporate a time period t in the challenge tuples, and the oracles are redefined to execute $\mathsf{Dec.U}$.

In the traceability experiment, the adversary now outputs a tuple $((epk_0, esk_0), (epk_1, esk_0), m)$, the challenger computes $\mathsf{Enc}(epk_\beta, pk_{\mathcal{T}}, m)$, and the adversary can request the challenger to run $\mathsf{Dec.U}$.

δ-correctness is not defined with respect to a time period anymore, and concerning recoverability, $\mathcal{A}$ need then not specify a time period $t > t_c$ in which B is δ-correct (equivalently, $\mathcal{A}$ could specify any dummy time period so long as t_c is set to $-\infty$), and algorithm B does not receive a time period t as an input.

The multi-user-case definitions are derived accordingly.

7.2 Generic Construction

Let Rand be a commitment re-randomization algorithm. Following the notation of Section 4, except that $\mathcal{E}_1$ is now an encryption scheme supporting labels (from every ciphertext of which the corresponding label can be efficiently computed), let $\mathcal{E}$ be an LDE scheme such that

$\mathsf{KeyEn} = (\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}}), \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})) \to ((epk, esk), epk)$: is a protocol between $\mathsf{KeyEn.U}$ and $\mathsf{KeyEn.A}$, which proceed as follows.

1. $\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}})$ and $\mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})$ run the interactive protocol $\mathsf{PoK}\{(s, o)\colon \mathsf{Open}(ck, c, s, o) = 1\}$ as prover and verifier respectively. If the protocol fails, the overall protocol is aborted, i.e., $epk = esk \leftarrow \perp$; otherwise
2. both algorithms set $epk = (pk_{\mathcal{U}}, c)$, and $\mathsf{KeyEn.U}$ sets $esk = (sk_{\mathcal{U}}, c, o)$

$\mathsf{Enc}(epk, pk_{\mathcal{T}}, m \in \mathcal{M}) \to ct$: generates $m_1 \in_R \mathcal{M}$, sets $m_0 = m \oplus m_1$, and outputs $ct = (\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, m_0), \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, m_1, l = \mathsf{Rand}(ck, c; r)), r)$

$\mathsf{Dec} = (\mathsf{Dec.U}(esk, ct), \mathsf{Dec.T}(sk_{\mathcal{T}}, ck))$: is an interactive protocol between $\mathsf{Dec.U}$ and $\mathsf{Dec.T}$ which proceed as follows.

1. Algorithm $\mathsf{Dec.U}$ sends (ct_1, l) to $\mathsf{Dec.T}$
2. algorithms $\mathsf{Dec.U}$ and $\mathsf{Dec.T}$ run protocol

$$\mathsf{PoK}\{(c, s, o, r)\colon \mathsf{Open}(ck, c, s, o) = 1,\ l = \mathsf{Rand}(ck, c; r)\}$$

   as prover and verifier respectively. If the protocol fails, $\mathsf{Dec.U}$ outputs $\perp$; otherwise
3. $\mathsf{Dec.T}$ sends $m_1 = \mathcal{E}_1.\mathsf{Dec}(sk_{\mathcal{T}}, ct_1, l)$ to $\mathsf{Dec.U}$; and
4. $\mathsf{Dec.U}$ outputs $\mathcal{E}_0.\mathsf{Enc}(sk, ct_0) \ominus m_1$ if $m_1 \neq \perp$ (outputs $\perp$ otherwise).

The other algorithms remain the same up to the omission of time periods. Observe that $\mathcal{E}_2$ and $\mathcal{S}$ are not used since the user need not encrypt her identity and prove that she knows a signature on it.

Similar proofs to those of Section 4 entail that $\mathcal{E}$ is correct if $\mathcal{E}_0$ and $\mathcal{E}_1$ are correct and if PoK is complete; that it satisfies privacy if $\mathcal{C}$ is hiding, and protocol PoK is ZK; that $\mathcal{E}$ is LD-IND-CCA secure if $\mathcal{E}_0$ is IND-CCA secure; and that untraceability is also satisfied if PoK is ZK. Scheme $\mathcal{E}$ also satisfies recoverability with respect to the class of distributions $\mathcal{D}$ such that $\delta > p(\mathcal{D})$ and $\delta - p(\mathcal{D})$ is non-negligible assuming $\mathcal{C}$ to be binding and $\mathcal{E}_1$ to be secure against chosen-ciphertext attacks [12, Section 2.4]. To prove it, it suffices to prove, as in Theorem 5 that with non-negligible probability, a δ-correct algorithm B sends the (third-party part of the) ciphertext and the label that it was given, and succeeds in the subsequent proof of knowledge. The knowledge extractor can then be used to retrieve the user secret.

7.3 Instantiation

Set $\mathcal{C}$ as the standard Pedersen commitment scheme and Rand as the algorithm that generates $r \in_r \mathbb{Z}_p$, and maps $c = g_1^s g_2^o$ to $c g_3^r$ for pairwise distinct g_1, g_2 and g_3. Let $\mathcal{E}_0$ be the Cramer–Shoup encryption scheme, and $\mathcal{E}_1$ be

the Cramer–Shoup encryption scheme supporting labels (Appendix A.1). Both schemes may have any group (in which the DDH assumption holds) with short representation as a common message space. The resulting ciphertexts are then composed of 9 group elements (one of which is a re-randomized commitment as a label), and decryption only requires exponentiations and two hash computations.

8 Revocation

Besides the scenario in which a decryption box was intentionally distributed by a user, the scenario in which a decryption device was stolen from an honest user or in which a user simply lost it is also relevant. In this case, the user should be able to prevent misuse of her device, i.e., unauthorized decryption. This can be achieve by adding a revocation functionality to LDE schemes. The interaction with the third party allows to easily add such a functionality: the third party need only also verify that the user's public key was not revoked without putting her anonymity at risk. Camenisch et al. [11] proposed a generic key-revocation component, which can be added to any system to enable a privacy-preserving revocation functionality. It is referred to as an Anonymous Revocation Component (ARC).

An ARC requires an entity called Revocation Authority (RA). In the present case, the RA can be the key-enhancement authority itself, the decryption third party or any other party in the system. The RA partakes in the key-enhancement protocol, maintains some revocation information, and changes the revocation status of the enhanced public keys.

Camenisch et al. [11, Section 4.4] provided definitions and a description of interfaces of an ARC, and instantiated it with the revocation scheme of Nakanishi et al. [23]. Baldimtsi et al. [3] gave an instance with accumulators. Both those instances are suitable for LDE schemes.

We first recall the definition of an ARC as proposed by Camenisch et al. [11], then we show how to add an ARC to our constructions without compromising the privacy of users' secrets or their anonymity.

8.1 Anonymous Revocation Components

In an ARC, revocation is achieved via a *revocation handle* $rh \in \mathcal{RH}$ that is embedded into the key to be revoked. An ARC $\mathcal{ARC}$ is a tuple of algorithms $(\mathsf{SPGen}, \mathsf{RKGen}, \mathsf{Revoke}, \mathsf{RevTokenGen}, \mathsf{RevTokenVer})$ such that

$\mathsf{SPGen}(1^\lambda) \to \mathcal{PP}_r$: is an algorithm that generates revocation parameters $\mathcal{PP}_r$

$\mathsf{RKGen}(\mathcal{PP}_r) \to (rpk, rsk, RI)$: is an algorithm that generates the RA's secret and public keys, and an initial revocation information RI

$\mathsf{Revoke}(rsk, RI, rh) \to RI'$: is an algorithm that revokes rh, and outputs an update RI' of the revocation information RI

$\mathsf{RevTokenGen}(rpk, RI, c, rh, o) \to rt$: is an algorithm which generates a publicly-verifiable revocation token proving that handle rh has not been revoked and that c is a commitment to rh

$\mathsf{RevTokenVer}(rpk, RI, c, rt) \to \{0,1\}$: is a revocation token verification algorithm.

The security requirements of an ARC can be informally stated as follows. *Correctness* states that any revocation token rt generated with a non-revoked handle rh and an honestly computed information RI is accepted by $\mathsf{RevTokenVer}$. *Soundness* ensures that $\mathsf{RevTokenVer}$ accepts rt and c on input RI and rpk only if rt was computed with rh and a valid opening o to c. Moreover, no party other than the RA can publish a valid revocation information RI, i.e., it is always authentic. *Revocation privacy* guarantees that given a revocation token rt, no information about the underlying revocation handle rh can be inferred.

8.2 Revocable (TB-)LDE Schemes

To add an ARC to the generic constructions of Sections 4, 6 and 7, it suffices to have the RA – recall that it partakes in the key-enhancement protocol – assign to each user, in addition to her secret, a revocation handle. In the case of TB-LDE schemes, this handle is signed by the key-enhancement authority together with the user's identity and commitment to her secret. To revoke a key, the corresponding handle is added to the publicly available revocation information. In (non–time-based) LDE schemes, the handle is also included in the computation of the label. During the key-derivation protocol for TB-LDE schemes or the decryption protocol for LDE schemes, the user computes a fresh (to be untraceable) commitment to the handle of her enhanced public key, and a revocation token, both sent to the third party. In addition to the proofs of those constructions, the user also proves that the handle of which she just sent a fresh commitment is not revoked, and that it was signed with the encrypted identity (for TB-LDE schemes) or used for the computation of the label (for LDE schemes).

Generic Construction of a Revocable TB-LDE Scheme. Let $\mathcal{E}$ be either of the Section-4 or the Section-6 TB-LDE scheme. Consider $\mathcal{E}_R$, a revocable TB-LDE scheme that has the same algorithms as $\mathcal{E}$, except for $\mathcal{E}_R.\mathsf{Setup}(1^\lambda)$ which also runs $\mathcal{ARC}.\mathsf{SPGen}(1^\lambda)$, for an additional algorithm $\mathcal{E}_R.\mathsf{Revoke} = \mathcal{ARC}.\mathsf{Revoke}$, and for its KeyEn and KeyDer protocols. Assuming, without loss of generality, the key-enhancement authority to also be the RA, those protocols are now

$\mathsf{KeyEn} = (\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}}), \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})) \to ((epk, esk), epk)$: a protocol between $\mathsf{KeyEn.U}$ and $\mathsf{KeyEn.T}$, which proceed as follows.

1. $\mathsf{KeyEn.U}$ and $\mathsf{KeyEn.T}$ run protocol $\mathsf{PoK}\{(s,o)\colon \mathsf{Open}(ck, c, s, o) = 1\}$ as prover and verifier respectively. If it fails, the overall key-enhancement protocol is aborted, i.e., $epk = esk \leftarrow \perp$; otherwise
2. $\mathsf{KeyEn.A}$ generates $(sk, vk) \leftarrow \mathcal{S}.\mathsf{KeyGen}(\mathcal{PP})$, $id \in_R \mathcal{ID}$, $rh \in_R \mathcal{RH}$, computes $\sigma = \mathsf{Sign}(sk, (c, id, rh))$, sets $epk = (pk_{\mathcal{U}}, c, id, rh, \sigma)$, and sends id and σ to $\mathsf{KeyEn.U}$, which sets epk as $\mathsf{KeyEn.A}$ and $esk = (sk_{\mathcal{U}}, c, s, o, id, rh, \sigma)$

$\mathsf{KeyDer} = (\mathsf{KeyDer.U}(esk, t), \mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck, vk, rpk, RI)) \rightarrow (sk_{\mathcal{T}}^{t,id}, \perp)$: a protocol between $\mathsf{KeyDer.U}$ and $\mathsf{KeyDer.T}$, which proceed as follows.

1. Algorithm $\mathsf{KeyDer.U}$ generates and stores a pair of encryption keys $(ek, dk) \leftarrow \mathcal{E}_2.\mathsf{KeyGen}(\mathcal{PP})$ if none was priorly stored, and otherwise reuses such a pair, computes a ciphertext $ct^{id} = \mathcal{E}_2.\mathsf{Enc}(ek, id; r_{\mathcal{U}})$, a commitment and an opening $(c_{rh}, o_{rh}) = \mathsf{Com}(ck, rh)$, and a revocation token $rt = \mathcal{ARC}.\mathsf{RevTokenGen}(rpk, RI, c_{rh}, rt)$, and sends $(ek, t, ct^{id}, c_{rh}, rt)$ to algorithm $\mathsf{KeyDer.T}$
2. if $t > t_c$, then $\mathsf{KeyDer.T}$ returns $\perp$, and $\mathsf{KeyDer.U}$ outputs $\perp$; otherwise, algorithms $\mathsf{KeyDer.U}$ and $\mathsf{KeyDer.T}$ run protocol

$$\mathsf{PoK}\{(c,s,o,id,rh,o_{rh},\sigma,r_{\mathcal{U}})\colon \mathsf{Open}(ck,c,s,o)=1,\ \mathsf{Open}(ck,c_{rh},rh,o_{rh})=1,\ \mathsf{Verify}(vk,(c,id,rh),\sigma)=1,\ ct^{id}=\mathcal{E}_2.\mathsf{Enc}(ek,id;r_{\mathcal{U}}),\ \mathsf{RevTokenGen}(rpk,RI,c_{rh},rh,o_{rh})=1\}$$

as prover and verifier respectively. If protocol PoK fails, the overall protocol is aborted, i.e., $sk_{\mathcal{T}}^{t,id} \leftarrow \perp$; otherwise

3. $\mathsf{KeyDer.T}$ computes $sk_{\mathcal{T}}'^{t,id} \leftarrow \mathsf{Der}_0(sk_{\mathcal{T}}, t, ek, ct^{id})$, and sends it to $\mathsf{KeyDer.U}$. Finally, $\mathsf{KeyDer.U}$ outputs $sk_{\mathcal{T}}^{t,id} = \mathsf{Der}_1(dk, sk_{\mathcal{T}}'^{t,id})$.

Generic Construction of a Revocable LDE Scheme. Assume $\mathcal{C}$ to be homomorphic, and Rand to be a commitment re-randomization algorithm. For $(c, o) = \mathsf{Com}(ck, m)$, algorithm $\mathsf{Rand}(ck, c, m'; r)$ computes a commitment to m and m' from c using the homomorphic property of $\mathcal{C}$, and re-randomizes the resulting commitment with randomness r. Let $\mathcal{E}$ be the Section-7 LDE scheme. Let $\mathcal{E}_R$ be a revocable LDE scheme, based on $\mathcal{E}$, with $\mathcal{E}_R.\mathsf{Setup}(1^\lambda)$ which also runs $\mathcal{ARC}.\mathsf{SPGen}(1^\lambda)$, and with an additional algorithm $\mathcal{E}_R.\mathsf{Revoke} = \mathcal{ARC}.\mathsf{Revoke}$, such that

$\mathsf{KeyEn} = (\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}}), \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})) \rightarrow ((epk, esk), epk)$: is a protocol between $\mathsf{KeyEn.U}$ and $\mathsf{KeyEn.T}$, which proceed as follows.

1. $\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}})$ and $\mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})$ run the interactive protocol $\mathsf{PoK}\{(s, o)\colon \mathsf{Open}(ck, c, s, o) = 1\}$ as prover and verifier respectively. If the proof fails, then the overall protocol is aborted, i.e., $epk = esk \leftarrow \perp$; otherwise
2. $\mathsf{KeyEn.A}$ generates $rh \in_R \mathcal{RH}$, and both algorithms set $epk = (pk_{\mathcal{U}}, c, rh)$, and $\mathsf{KeyEn.U}$ sets $esk = (sk_{\mathcal{U}}, c, o)$.

$\mathsf{Enc}(epk, pk_{\mathcal{T}}, m \in \mathcal{M}) \rightarrow ct$: generates $m_1 \in_R \mathcal{M}$, sets $m_0 = m \oplus m_1$, and outputs $ct = (\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, m_0), \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, m_1, l = \mathsf{Rand}(ck, c, rh; r)), r)$

$\mathsf{Dec} = (\mathsf{Dec.U}(esk, ct), \mathsf{Dec.T}(sk_{\mathcal{T}}, ck)) \rightarrow (m, \perp)$: is an interactive protocol between $\mathsf{Dec.U}$ and $\mathsf{Dec.T}$ which proceed as follows.

1. Algorithm $\mathsf{Dec.U}$ computes $(c_{rh}, o_{rh}) = \mathsf{Com}(ck, rh)$, a revocation token $rt = \mathcal{ARC}.\mathsf{RevTokenGen}(rpk, RI, c_{rh}, rt)$, and sends (ct_1, l, c_{rh}, rt) to $\mathsf{Dec.T}$

2. algorithms Dec.U and Dec.T run protocol

$$\mathsf{PoK}\{(c,s,o,c_{rh},rh,o_{rh},r)\colon \mathsf{Open}(ck,c,s,o)=1,\ \mathsf{Open}(ck,c_{rh},rh,o_{rh})=1,\\ l=\mathsf{Rand}(ck,c,rh;r),\ \mathsf{RevTokenGen}(rpk,RI,c_{rh},rh,o_{rh})\}$$

as prover and verifier respectively. If it fails, Dec.U outputs $\perp$; otherwise
3. Dec.T sends $m_1=\mathcal{E}_1.\mathsf{Dec}(sk_{\mathcal{T}},ct_1,l)$ to Dec.U
4. Dec.U outputs $\mathcal{E}_0.\mathsf{Enc}(sk,ct_0)\ominus m_1$ if $m_1\neq\perp$ (outputs $\perp$ otherwise).

8.3 Security of Revocable (TB-)LDE-Schemes

The correctness of a revocable (TB-)LDE scheme should now take the ARC into account, i.e., state that the decryption of the encryption of a message should result in the message itself if the public key is not revoked. The other security requirements remain the same. Yet, to ensure that ciphertexts cannot be decrypted after the public keys used to compute them have been revoked, an additional property must be introduced: *revocation soundness*.

Definition 6 (Revocation Soundness of a TB-LDE Scheme). *A revocable TB-LDE scheme $\mathcal{E}_R$ satisfies revocation soundness if there exists a negligible function such that for $(\mathcal{PP},ck)\leftarrow\mathsf{Setup}(1^\lambda,aux)$, for all $(sk,vk)\leftarrow\mathcal{S}.\mathsf{KeyGen}(\mathcal{PP})$, $(rpk,rsk,RI)\leftarrow\mathcal{ARC}.\mathsf{RKGen}(\mathcal{PP}_r)$, for every pair of truthfully enhanced keys $(epk=(pk_{\mathcal{U}},c,id,rh,\sigma),esk)$, for every truthfully generated third-party secret key $sk_{\mathcal{T}}\in\mathcal{R}_{\mathcal{T}}$, for every message $m\in\mathcal{M}$ and every time period t,*

$$\Pr[\mathcal{ARC}.\mathsf{Revoke}(rsk,RI,rh)=RI',\ (\mathsf{KeyDer.U}(esk,t),\\ \mathsf{KeyDer.T}(sk_{\mathcal{T}},ck,vk,rpk,RI'))\neq(\perp,\perp)]\leq\mathrm{negl}(\lambda).$$

The revocation soundness of an LDE Scheme is defined analogously with Dec and plaintexts.

Regarding the Section-8.2 construction, the incorporation of an ARC clearly does not affect the privacy, the LD-IND-CCA security or the recoverability of $\mathcal{E}$, be it a TB-LDE or a LDE scheme. In the case of an (TB-)LDE scheme, since the handle included in the computation of the label (signature), the argument for recoverability remains the same as that of Section 7 (4). However, untraceability may now be at risk due to the commitment to the revocation handle and the revocation token sent to the third party. Still, similar arguments to those of Theorem 4 imply that $\mathcal{E}_R$, based on a (TB-)LDE scheme $\mathcal{E}$, satisfies untraceability if protocol PoK is ZK (, $\mathcal{E}_2$ is IND-CPA secure) and $\mathcal{ARC}$ satisfies revocation privacy. Furthermore, $\mathcal{E}_R$, satisfies revocation soundness if $\mathcal{ARC}$ is sound (, $\mathcal{S}$ is existentially unforgeable) and protocol PoK is sound. Indeed, the unforgeability of $\mathcal{S}$ and the soundness of PoK imply that the handle used in the KeyDer is the one of the user enhanced public key. The soundness of $\mathcal{ARC}$ then implies that the handle was not revoked.

9 Interactive Recoverability

A key-revocation component only allows a user to prevent further use of her device in case of loss or theft[7]. Nevertheless, as the Section-4 Rec algorithm is not interactive, a user's secret can still be recovered by anyone in possession of her device even if she has requested to have her public key revoked. To also protect user secrets in case she lost her device, local recoverability must be prevented, i.e., recoverability without interaction with the decryption third party $\mathcal{T}$ which checks that user public keys are not revoked.

To enforce interaction with $\mathcal{T}$, it suffices to require that any decryption algorithm, interacting with any other algorithm that does not possess the third-party master secret key, has only a negligible probability to correctly decrypt ciphertexts in future time periods.

Definition 7 (Interaction Soundness). *A TB-LDE scheme $\mathcal{E}$ satisfies interaction soundness if for every efficient algorithm* B*, for all* $(epk, esk) \in \mathcal{ER}$*,* $(pk_{\mathcal{T}}, sk_{\mathcal{T}}) \in \mathcal{R}_{\mathcal{T}}$*, for every efficient algorithm* $\mathcal{A}(1^{\lambda})$*, there exists a negligible function* negl *such that for every time period* $t > t_c$*, for all* $m \in \mathcal{M}$*,*

$$\Pr\left[\mathsf{B}^{\mathcal{A}(epk,esk,pk_{\mathcal{T}},ck)}(\mathsf{Enc}(epk, pk_{\mathcal{T}}, m, t)) = m\right] \leq \mathrm{negl}\,(\lambda)\,.$$

In particular, no user decryption device, even one which runs KeyDer.U and Dec as subroutines, can correctly decrypt ciphertexts if it does not interact with the third party which holds $sk_{\mathcal{T}}$. That is to say, protocol PoK always fails, and its knowledge extractor $\mathcal{K}$ cannot be used to recover user secrets.

Interaction Soundness in the (TB-)LDE constructions. To add interaction soundness to the Section-7(4) (TB-)LDE scheme, it suffices to make the Dec.T (KeyDer.T) algorithm sign the first messages sent by Dec.U (KeyDer.U) and its challenges with an existentially unforgeable scheme $\mathcal{S}'$. Before answering the challenge, algorithm Dec.U (KeyDer.U) verifies the signature with the public verification key. It follows that unless an adversary, which does not possess the third-party signing key, can forge a signature, protocol PoK will never terminate, and algorithm Rec will never produce any output, and the user secret cannot be recovered. The existential unforgeability of $\mathcal{S}'$ then implies the interaction soundness of the (TB-)LDE scheme.

10 Conclusion

We first argued that in leakage-deterring schemes, a CCA type of privacy of user secrets is compatible with their recoverability. We therefore redefined leakage-deterring schemes with security guarantees stronger than existing ones. We then

[7] Notice that the loss of a decryption device does not mean the loss of the secret secret since the latter is simply embedded into the device. It might also be kept elsewhere, allowing the user to identify herself to the authority when she asks for her public key to be revoked.

gave a construction that turns any CCA-secure encryption scheme into a leakage deterring one that achieves those stronger guarantees and has constant-size ciphertexts in the size of user secrets.

The main drawback of our construction is the need to interact once per epoch (e.g., a week) with a party that helps users decrypt. However, this very same interaction is needed to guarantee a CCA type of privacy of user secrets together with their recoverability, and can even be leveraged to revoke user keys and protect their secrets in case or loss or theft.

Acknowledgements. This work supported by the ERC Grant PERCY #321310.

References

1. M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic k-TAA. Cryptology ePrint Archive, Report 2008/136, 2008. `http://eprint.iacr.org/2008/136`.
2. M. Backes, J. Müller-Quade, and D. Unruh. On the necessity of rewinding in secure multiparty computation. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 157–173. Springer, Heidelberg, Feb. 2007.
3. F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov. Accumulators with applications to anonymity-preserving revocation. EuroS&P, 2017.
4. B. Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, Oct. 2001.
5. D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, Apr. 2008.
6. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, Aug. 2004.
7. D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, Heidelberg, Aug. 1999.
8. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Heidelberg, May / June 2006.
9. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, Aug. 2014.
10. J. Camenisch, M. Drijvers, and A. Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. Cryptology ePrint Archive, Report 2016/663, 2016. `http://eprint.iacr.org/2016/663`.
11. J. Camenisch, S. Krenn, A. Lehmann, G. L. Mikkelsen, G. Neven, and M. Ø. Pedersen. Formal treatment of privacy-enhancing credential systems. In *SAC*, pages 3–24, 2015.
12. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Heidelberg, Aug. 2003.

13. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Heidelberg, Aug. 1994.
14. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, Aug. 1998.
15. C. Dwork, J. B. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information (preliminary version). In *28th ACM STOC*, pages 489–498. ACM Press, May 1996.
16. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, Oct. / Nov. 2006. Available as Cryptology ePrint Archive Report 2006/309.
17. V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *42nd FOCS*, pages 658–667. IEEE Computer Society Press, Oct. 2001.
18. A. Kiayias and Q. Tang. How to keep a secret: leakage deterring public-key cryptosystems. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 943–954. ACM Press, Nov. 2013.
19. A. Kiayias and Q. Tang. Traitor deterring schemes: Using bitcoin as collateral for digital content. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 231–242. ACM Press, Oct. 2015.
20. A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 450–465. Springer, Heidelberg, Apr. / May 2002.
21. K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 145–157. Springer, Heidelberg, May / June 1998.
22. A. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. Cryptology ePrint Archive, Report 2011/490, 2011. `http://eprint.iacr.org/2011/490`.
23. T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 463–480. Springer, Heidelberg, Mar. 2009.
24. R. Nishimaki, D. Wichs, and M. Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 388–419. Springer, Heidelberg, May 2016.
25. T. Okamoto and K. Takashima. Homomorphic encryption and signatures from vector decomposition. In S. D. Galbraith and K. G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 57–74. Springer, Heidelberg, Sept. 2008.
26. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, Aug. 2010.

A Preliminaries

This section gives instantiations for our construction building blocks.

A.1 Cramer–Shoup Encryption Scheme

The Cramer–Shoup encryption scheme [14] consists of the following algorithms:

$\mathsf{Setup}(1^\lambda) \to \mathcal{PP} = (p, \mathbb{G}, \mathscr{H})$: output a prime number $2^{\lambda-1} \leq p < 2^\lambda$, the description of a p-order group $\mathbb{G}$, and the description of family of universal one-way hash functions

$\mathsf{KeyGen}(\mathcal{PP}) \to (pk, sk)$: generate $x_1, x_2, y_1, y_2, z \in_R \mathbb{Z}_p, G_1, G_2 \in_R \mathbb{G}$, sets $A = G_1^{x_1} G_2^{x_2}, B = G_1^{y_1} G_2^{y_2}$ and $H = G_1^z$, choose $\mathsf{Hash} \in_R \mathscr{H}$, and output $pk = (G_1, G_2, A, B, H, \mathsf{Hash})$ and $sk = (x_1, x_2, y_1, y_2, z, \mathsf{Hash})$

$\mathsf{Enc}(pk, M \in \mathbb{G}) \to ct$: generate $r \in_R \mathbb{Z}_p$, compute $U_1 = G_1^r, U_2 = G_2^r, E = H^r M, \alpha = \mathsf{Hash}(U_1, U_2, E), V = (AB^\alpha)^r$, and output a ciphertext $ct = (U_1, U_2, E, V)$

$\mathsf{Dec}(sk, ct) \to M$: parse ct as (U_1, U_2, E, V) (outputs $M = \perp$ if not possible), compute $\alpha = \mathsf{Hash}(U_1, U_2, E)$, and outputs $M = E/U_1^z$ if $V = U_1^{x_1+\alpha y_1} U_2^{x_2+\alpha y_2}$, and otherwise output $\perp$.

It is secure against chosen-ciphertext attacks under the DDH assumption. The Cramer–Shoup encryption scheme can also support labels. A label need only be included in the hash and appended to the ciphertext. The security proof [14, Theorem 1, Lemma 2, Claim 2] of the scheme remains unchanged so long as $\mathscr{H}$ is a family of collision-resistant hash functions.

A.2 Dual Pairing Vector Spaces

Dual Pairing Vector Spaces (DPVSs) were introduced by Okamoto and Takashima [25]. They provide a mechanism for parameter hiding [22] in prime-order pairing groups. The latter feature allows to proves the full security of functional encryption schemes in prime-order settings.

Definition 8 (Dual Pairing Vector Space). *Let $N \geq 1$ be an integer. A dual pairing vector space by direct product of a pairing group $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e)$ is a tuple $(p, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, e)$ such that $\mathbb{V} = \mathbb{G}_1^N$ and $\mathbb{V}^* = \mathbb{G}_2^N$ are two N-dimensional $\mathbb{Z}_p$ vector spaces, $\mathbb{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ is the canonical basis of $\mathbb{V}$ (i.e., $\mathbf{a}_i = (\mathbf{1}_{\mathbb{G}_1^{i-1}}, P_1, \mathbf{1}_{\mathbb{G}_1^{N-i}})$), $\mathbb{A}^* = (\mathbf{a}_1^*, \ldots, \mathbf{a}_N^*)$ is the canonical basis of $\mathbb{V}^*$ (i.e., $\mathbf{a}_i^* = (\mathbf{1}_{\mathbb{G}_2^{i-1}}, P_2, \mathbf{1}_{\mathbb{G}_2^{N-i}})$) and*

$$e \colon \mathbb{V} \times \mathbb{V}^* \to \mathbb{G}_T$$
$$(\mathbf{x} = (X_1, \ldots, X_N), \mathbf{y} = (Y_1, \ldots, Y_N)) \mapsto \prod_i e(X_i, Y_i)$$

(note the abuse of notation) is a pairing, i.e., $\mathbf{x} = \mathbf{1}_{\mathbb{G}_1^N}$ if $e(\mathbf{x}, \cdot)$ is the $1_{\mathbb{G}_T}$ map, and $\forall a, b \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{V}, \mathbf{y} \in \mathbb{V}^*, e(\mathbf{x}^a, \mathbf{y}^b) = e(\mathbf{x}, \mathbf{y})^{ab}$.

Note that for all $1 \leq i, j \leq N, e(\mathbf{a}_i, \mathbf{a}_j^*) = e(P_1, P_2)^{\delta_{ij}}$, with δ_{ij} being the Kronecker delta, i.e., $\delta_{ij} = 1$ if $i = j$, and otherwise 0.

Let $\mathcal{G}_{\mathrm{dpvs}}(1^\lambda, \mathcal{PP}_\mathbb{G}, N)$ denote an algorithm that takes as an input a security parameter 1^λ, the description $\mathcal{PP}_\mathbb{G}$ of a pairing group $(p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e)$ and an integer N, and outputs the description $\mathcal{PP}_\mathbb{V}$ of a DPVS $(p, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, e)$.

A.3 Okamoto–Takashima Adaptively-Secure CP-ABE Scheme

Let $\mathcal{G}_{\mathrm{ob}}(1^\lambda, \mathbf{n} = (d; n_1, \ldots, n_d))$ be a dual-orthonormal-basis generator which proceeds as follows:

1. it generates a pairing group $\mathcal{PP}_{\mathbb{G}} = (p, \mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T, e) \leftarrow \mathcal{G}_{\mathrm{bpg}}(1^\lambda)$, a value $\psi \in_R \mathbb{Z}_p^*$, and sets $N_0 = 5$ and $N_k = 3n_k + 1$ for $1 \le k \le d$
2. for $0 \le k \le d$, it generates a Dual Pairing Vector Space (DPVS) (see Appendix A.2) $\mathcal{PP}_{\mathbb{V}_k} = (p, \mathbb{V}_k, \mathbb{V}_k^*, \mathbb{G}_T, \mathbb{A}_k, \mathbb{A}_k^*, e) \leftarrow \mathcal{G}_{\mathrm{dpvs}}(1^\lambda, \mathcal{PP}_{\mathbb{G}}, N_k)$, generates a matrix $\mathbf{X}_k \in_R \mathrm{GL}_{N_k}(\mathbb{Z}_p)$, and computes $\mathbf{V}_k = \psi \left(\mathbf{X}_k^{\mathrm{T}}\right)^{-1}$. Let $\mathbf{M}_{\mathbb{A}_k}$ and $\mathbf{M}_{\mathbb{A}_k^*}$ respectively denote the diagonal matrices $\mathrm{diag}(P_1) \in \mathbb{G}_1^{N_k \times N_k}$ and $\mathrm{diag}(P_2) \in \mathbb{G}_2^{N_k \times N_k}$. Generator $\mathcal{G}_{\mathrm{ob}}$ computes $\mathbf{B}_k = \begin{bmatrix} \mathbf{b}_{k,1} \\ \vdots \\ \mathbf{b}_{k,n} \end{bmatrix} = \mathbf{X}_k \mathbf{M}_{\mathbb{A}_k} \in \mathbb{G}_1^{N_k \times N_k}$ and $\mathbf{B}_k^* = \begin{bmatrix} \mathbf{b}_{k,1}^* \\ \vdots \\ \mathbf{b}_{k,n}^* \end{bmatrix} = \mathbf{V}_k \mathbf{M}_{\mathbb{A}_k^*} \in \mathbb{G}_2^{N_k \times N_k}$ with $\mathbf{b}_{k,i} = (\mathbf{e}_i \cdot \mathbf{X}_k)\, \mathbf{M}_{\mathbb{A}_k} = \left[P_1^{\mathbf{X}_{k,i,1}} \ \cdots \ P_1^{\mathbf{X}_{k,i,N_k}}\right]$ and $\mathbf{b}_{k,i}^* = (\mathbf{e}_i \cdot \mathbf{V}_k)\, \mathbf{M}_{\mathbb{A}_k^*} = \left[P_2^{\mathbf{V}_{k,i,1}} \ \cdots \ P_2^{\mathbf{V}_{k,i,N_k}}\right]$
3. it computes $G_T = e(P_1, P_2)^\psi$, sets $\mathcal{PP}_{\mathbf{n}} = \{\mathcal{PP}_{\mathbb{V}_k}\}_{0 \le k \le d}$, and eventually outputs $(G_T, \mathcal{PP}_{\mathbf{n}}, \{\mathbf{B}_k, \mathbf{B}_k^*\}_{0 \le k \le d})$.

Notice that for all i, k, $G_T = e(\mathbf{b}_{k,i}, \mathbf{b}_{k,i}^*)$. Indeed,

$$\begin{aligned} e(\mathbf{b}_{k,i}, \mathbf{b}_{k,i}^*) &= e\left(\prod_j \mathbf{a}_j^{\mathbf{X}_{ij}}, \prod_l \mathbf{a}_l^{*\mathbf{V}_{il}}\right) = \prod_{j,l} e(\mathbf{a}_j, \mathbf{a}_l^*)^{\mathbf{X}_{ij}\mathbf{V}_{il}} \\ &= \prod_{j,l} e(P_1, P_2)^{\delta_{jl}\mathbf{X}_{ij}\mathbf{V}_{il}} = e(P_1, P_2)^\psi = G_T. \end{aligned}$$

Consider now the (monotone-span-program) Okamato–Takashima CP-ABE scheme [26, Section 7.1] in the case $d = 2$. The access structure associated to a ciphertext is determined by two 2-dimensional vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. A pair of attributes (a pair of $\mathbb{Z}_p$-lines) represented by a pair of vectors $(\mathbf{x}_1, \mathbf{x}_2)$ is "accepted" by the structure if and only if $\mathbf{x}_k \cdot \mathbf{v}_k^{\mathrm{T}} = 0$: that is, the structure specifies two accepted $\mathbb{Z}_p$-lines. Their CP-ABE scheme is defined as follows:

$\mathsf{Setup}(1^\lambda, \mathbf{n} = (2; n_1 = 2, n_2 = 2)) \rightarrow (pk, msk)$: generate an orthonormal basis $(G_T, \mathcal{PP}_{\mathbf{n}}, \{\mathbf{B}_k, \mathbf{B}_k^*\}_{0 \le k \le 2}) \leftarrow \mathcal{G}_{\mathrm{ob}}(1^\lambda, \mathbf{n})$, set $\hat{\mathbf{B}}_0 = \begin{bmatrix} \mathbf{b}_{0,1} \\ \mathbf{b}_{0,3} \\ \mathbf{b}_{0,5} \end{bmatrix}, \hat{\mathbf{B}}_0^* = \begin{bmatrix} \mathbf{b}_{0,1}^* \\ \mathbf{b}_{0,3}^* \\ \mathbf{b}_{0,4}^* \end{bmatrix}$, $\hat{\mathbf{B}}_k = \begin{bmatrix} \mathbf{b}_{k,1} \\ \mathbf{b}_{k,2} \\ \mathbf{b}_{k,7} \end{bmatrix}, \hat{\mathbf{B}}_k^* = \begin{bmatrix} \mathbf{b}_{k,1}^* \\ \mathbf{b}_{k,2}^* \\ \mathbf{b}_{k,5}^* \\ \mathbf{b}_{k,6}^* \end{bmatrix}$ for $k = 1, 2$, and then output $pk = (G_T, \mathcal{PP}_{\mathbf{n}}, \{\hat{\mathbf{B}}_k\}_{k=0,\ldots,2}), msk = (pk, \{\hat{\mathbf{B}}_k^*\}_{k=0,\ldots,2})$

$\mathsf{KeyDer}(msk, \mathcal{A} = \{\mathbf{x}_{k=1,2} \in \mathbb{Z}_p^2 \colon \mathbf{x}_{k,1} = 1\}) \to sk_{\mathcal{A}}$: generate $\alpha, y_0 \in_R \mathbb{Z}_p$, $\mathbf{y}_k \in_R \mathbb{Z}_p^2$ for $k = 1, 2$, compute vectors $\mathbf{k}_0^* = \begin{bmatrix}\alpha\ 0\ 1\ y_0\ 0\end{bmatrix} \mathbf{B}_0^* = \begin{bmatrix}\alpha\ 1\ y_0\end{bmatrix} \hat{\mathbf{B}}_0^*$, $\mathbf{k}_k^* = \begin{bmatrix}\alpha\mathbf{x}_k\ \mathbf{0}_{\mathbb{Z}_p^2}\ \mathbf{y}_k\ 0\end{bmatrix} \mathbf{B}_k^* = \begin{bmatrix}\alpha\mathbf{x}_k\ \mathbf{y}_k\end{bmatrix} \hat{\mathbf{B}}_k^*$, and output secret key $sk_{\mathcal{A}} = (pk, \mathcal{A}, \{\mathbf{k}_k^*\}_{k=0,\ldots,2})$

$\mathsf{Enc}(pk, M \in \mathbb{G}_T, \mathbb{S} = (\mathbf{v}_1, \mathbf{v}_2)) \to ct$: generate uniformly random values $a_1, a_2, \zeta, \eta_0, \eta_k, \theta_k$ for $k = 1, 2$ from $\mathbb{Z}_p$, computes $a = a_1 + a_2$,

$$\begin{aligned}
\mathbf{c}_0 &= \begin{bmatrix}-a\ 0\ \zeta\ 0\ \eta_0\end{bmatrix} \mathbf{B}_0 = \begin{bmatrix}-a\ \zeta\ \eta_0\end{bmatrix} \hat{\mathbf{B}}_0, \\
\mathbf{c}_k &= \begin{bmatrix}a_k\mathbf{e}_{k,1} + \theta_k\mathbf{v}_k\ \mathbf{0}_{\mathbb{Z}_p^4}\ \eta_1\end{bmatrix} \mathbf{B}_k = \begin{bmatrix}a_k\mathbf{e}_{k,1} + \theta_k\mathbf{v}_k\ \eta_1\end{bmatrix} \hat{\mathbf{B}}_k \quad \text{for } k = 1, 2, \\
c_3 &= G_T^{\zeta} M,
\end{aligned}$$

and output $ct = (\mathbb{S}, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, c_3) \in \mathbb{Z}_p^4 \times \mathbb{G}_1^{19} \times \mathbb{G}_T$ and

$\mathsf{Dec}(sk_{\mathcal{A}}, ct) \to M$: output $M = c_3 / e(\mathbf{c}_0, \mathbf{k}_0^*) e(\mathbf{c}_1, \mathbf{k}_1^*) e(\mathbf{c}_2, \mathbf{k}_2^*)$ if the key and the ciphertext can be properly parsed and $\mathbf{x}_k \cdot \mathbf{v}_k^{\mathrm{T}} = 0 \mod p$ for $k = 1, 2$, and otherwise output $\perp$.

Since attribute vectors have their first coordinates set to 1, (the second coordinates specify the slopes of the $\mathbb{Z}_p$-lines accepted by the access structure), the attribute set may be identified with $\mathbb{Z}_p^2$. Okamoto and Takashima proved that this CP-ABE scheme is correct and adaptively payload-hiding against chosen-message attacks under the DLIN assumption over $\mathbb{G}_1$ and $\mathbb{G}_2$ [26, Theorem 2].

B Definitions and Security Model

This section gives the correctness definition omitted in the body of the paper as well as definitions in the multi-user case.

Definition 9 (Correctness). *$\mathcal{E}$ is correct if $\forall \lambda \in \mathbb{N}$, $\forall aux$,*

1. $(\mathcal{PP}, ck) \leftarrow \mathsf{Setup}(1^\lambda, aux)$,
2. $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) \leftarrow \mathsf{KeyGen.U}(\mathcal{PP})$, $(pk_{\mathcal{T}}, sk_{\mathcal{T}}) \leftarrow \mathsf{KeyGen.T}(\mathcal{PP})$,
3. $\forall (c, s, o)$ *such that* $\mathsf{Open}(ck, c, s, o) = 1$,

$$((epk, esk), epk) \leftarrow (\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, sk_{\mathcal{U}}), \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}}))\,,$$

4. $\forall t$, $(sk_{\mathcal{T}}^{t,\mathcal{U}}, \perp) \leftarrow (\mathsf{KeyDer.U}(esk, t), \mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck))$, $\forall m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(esk, sk_{\mathcal{T}}^{t,\mathcal{U}}, \mathsf{Enc}(epk, pk_{\mathcal{T}}, m, t)) = m\right] = 1.$$

B.1 Security Definitions in the Multi-User Case

As for traitor-deterring schemes [19], the previous definitions can be naturally extended to a multi-user case. The secrecy of a user's secrets and messages, and her untraceability are now also jeopardized by the other users. As for recoverability, several users may collude to produce a pirate decryption device.

In the presence of n users, the key-enhancement protocol of the single-user case is executed once for each of them. The encryption and decryption algorithms remain the same as those of the single-user case. The recovery algorithm takes as an input an algorithm B, a set $\mathcal{J} \subseteq \{1, \ldots, n\}$ of colluding users, $|\mathcal{J}|$ enhanced public keys, the description of $|\mathcal{J}|$ distributions, $|\mathcal{J}|$ time periods and a third-party public key $pk_{\mathcal{T}}$.

Correctness is defined, for each user, as in the single user case. Concerning δ-correctness, an algorithm B is said to be δ-correct in time periods $t_{i_1}, \ldots, t_{i_{|\mathcal{J}|}}$ with respect to distributions $\mathcal{D}_{i_1}, \ldots, \mathcal{D}_{i_{|\mathcal{J}|}}$ if for all $1 \leq j \leq |\mathcal{J}|$, $m_{i_j} \leftarrow\!\!\$\, \mathcal{D}_{i_j}$, $\Pr\left[\mathsf{B}^{\mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck, t_{i_j})}(\mathsf{Enc}(epk_{i_j}, pk_{\mathcal{T}}, m_{i_j}, t_{i_j})) = m_{i_j}\right] \gtrsim \delta$.

The privacy, LD-IND-CCA and untraceability definitions remain the same as they ought to guarantee the secrecy of any user secret, messages and identity, even if the other users, the third party and the authority are malicious and collude.

As to recoverability, a multi-user TB-LDE scheme satisfies recoverability with respect to distribution classes $\mathscr{D}_{i_1}, \ldots, \mathscr{D}_{i_{|\mathcal{J}|}}$ if for every efficient adversary $\mathcal{A}(1^\lambda)$, there exists a negligible function negl such that

$$\Pr\Big[\ s' \notin \{s_{i_j}\}_{j \in |\mathcal{J}|}, \mathcal{D}_{i_j} \in \mathscr{D}_{i_j}, t_{i_j} > t_c, \mathsf{B}\ \delta\text{-correct in } t_{i_j} \text{ w.r.t. } \mathcal{D}_{i_j} : \\ \left.\begin{array}{c} (\mathcal{PP}, ck) \leftarrow \mathsf{Setup}(1^\lambda),\ (pk_{\mathcal{T}}, sk_{\mathcal{T}}) \leftarrow \mathsf{KeyGen.T}(\mathcal{PP}), \\ (s_1, \ldots, s_n, pk_{\mathcal{U}_1}, \ldots, pk_{\mathcal{U}_n}) \leftarrow \mathcal{A}(\mathcal{PP}, ck, pk_{\mathcal{T}}),\ (c_i, o_i) \leftarrow \mathsf{Com}(ck, s_i)_{1 \leq i \leq n}, \\ (\cdot, epk_i)_{1 \leq i \leq n} \leftarrow (\mathcal{A}(c_1, o_1, \ldots, c_n, o_n),\ \mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}_i}))_{1 \leq i \leq n}, \\ (\mathsf{B}, (\mathcal{D}_{i_j}, t_{i_j})_{j \in \mathcal{J}}) \leftarrow \mathcal{A}^{\mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck, t_{i_j})},\ s' \leftarrow \mathsf{Rec}(\mathsf{B}, \mathcal{J}, (epk_{i_j}, \mathcal{D}_{i_j}, t_{i_j})_{j \in \mathcal{J}}, pk_{\mathcal{T}}) \end{array}\right] \\ \leq \mathrm{negl}\,(\lambda)\,.$$

Note that this definition allows each of the colluding users to specify a distribution of messages of which the ciphertexts are accepted by B. Indeed, colluding users may for instance want the implementation to decipher emails coming from different senders. In this sense, this definition is broader than Kiayias and Tang's for traitor-deterring schemes [19].

C Generic Construction of a TB-LDE Scheme

This sections gives the proofs of security of the construction in Section 4.

Proof (of Theorem 2 – Privacy). Let $\mathcal{A}$ be an adversary for the privacy distinction experiment. Consider an algorithm $\mathcal{S}$, which interacts with $\mathcal{A}$ and a commitment-scheme hiding-experiment challenger $\mathcal{C}_\beta$ that always commits to s_β for $\beta \in \{0, 1\}$. After receiving a commitment key ck from $\mathcal{C}_\beta$, algorithm $\mathcal{S}$ runs $\mathcal{E}.\mathsf{Setup}(1^\lambda)$, and generates a pair of keys $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$. Algorithm $\mathcal{S}$ then sends $(ck, pk_{\mathcal{U}})$ to $\mathcal{A}$. Upon receiving a pair (s_0, s_1) from $\mathcal{A}$, algorithm $\mathcal{S}$ forwards it to $\mathcal{C}_\beta$, and gets back c, a commitment to s_β. Algorithm $\mathcal{S}$ sends c to $\mathcal{A}$, and simulates, in protocol KeyEn, a proof of knowledge of a secret and of an opening to c by calling on the simulator of proof system PoK. Thereafter, whenever $\mathcal{A}$ issues a key-derivation query, algorithm $\mathcal{S}$, in protocol KeyDer, generates

a pair of keys $(ek, dk) \leftarrow \mathcal{E}_2.\mathsf{KeyGen}(\mathcal{PP})$ and an identity $id' \in_R \mathcal{ID}$, computes $ct^{id'} = \mathcal{E}_2.\mathsf{Enc}(ek, id')$, and sends ek, t and $ct^{id'}$ to $\mathcal{A}$. It then simulates a proof of knowledge using the simulator of proof system PoK. Algorithm $\mathcal{S}$ ultimately forwards the decision bit of $\mathcal{A}$ to $\mathcal{C}_\beta$. Note that

$$\begin{aligned}
&\left|\Pr\left[\mathbf{Exp}^{\text{priv}-0}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{priv}-1}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right]\right| \leq \\
&\left|\Pr\left[\mathbf{Exp}^{\text{priv}-0}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{commit}-0}_{\mathcal{C},\lambda}(\mathcal{S}(\mathcal{A})) = 1\right]\right| \\
&+ \left|\Pr\left[\mathbf{Exp}^{\text{commit}-0}_{\mathcal{C},\lambda}(\mathcal{S}(\mathcal{A})) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{commit}-1}_{\mathcal{C},\lambda}(\mathcal{S}(\mathcal{A})) = 1\right]\right| \\
&+ \left|\Pr\left[\mathbf{Exp}^{\text{commit}-1}_{\mathcal{C},\lambda}(\mathcal{S}(\mathcal{A})) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{priv}-1}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right]\right|.
\end{aligned}$$

The first and third terms are negligible if proof system PoK is ZK and $\mathcal{E}_2$ is IND-CPA secure. The second term is negligible (or rather nil) if $\mathcal{C}$ is (perfectly) hiding. □

Proof (of Theorem 3 – LD-IND-CCA Security). We prove that the LD-IND-CCA security of $\mathcal{E}$ can be reduced to the IND-CCA security of $\mathcal{E}_0$. Indeed, if $\mathcal{A}$ is an adversary for the LD-IND-CCA security game, consider $\mathcal{S}$, an algorithm which interacts with $\mathcal{A}$ and the challenger $\mathcal{C}$ of the IND-CCA security game. Upon receiving a public key $pk_{\mathcal{U}}$ from $\mathcal{C}$, algorithm $\mathcal{S}$ generates a commitment key ck, and forwards $pk_{\mathcal{U}}$ and ck to $\mathcal{A}$. When $\mathcal{A}$ sends a commitment c, a secret s and an opening o to $\mathcal{S}$, the latter runs $\mathsf{KeyEn.U}(ck, c, s, o, pk_{\mathcal{U}}, \perp)$. If the protocol terminates, the enhanced public key of $\mathcal{S}$ is set to $(pk_{\mathcal{U}}, c, id, \sigma)$ for an identity id and a signature σ generated by $\mathcal{A}$. Afterwards, whenever $\mathcal{A}$ requests the decryption of a ciphertext, $\mathcal{S}$ forwards its first part to $\mathcal{C}$, and can subsequently answer the request. Key-derivation queries can be directly answered by $\mathcal{S}$ as $sk_{\mathcal{U}}$ is not needed in the protocol. Upon receiving a challenge tuple $(m_0, m_1, t, pk_{\mathcal{T}})$ from $\mathcal{A}$, algorithm $\mathcal{S}$ generates $m \in_R \mathcal{M}$, sends $(m \oplus m_0, m \oplus m_1)$ to $\mathcal{C}$, gets back a challenge ciphertext ct^*, computes $ct_1 = \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, m, \{t, id\})$, and sends (ct^*, ct_1) to $\mathcal{A}$. Whenever $\mathcal{A}$ requests the decryption of a ciphertext $ct = (ct_0, ct_1)$ such that $ct_0 \neq ct^*$, algorithm $\mathcal{S}$ forwards ct_0 to $\mathcal{C}$, and then follows the rest of the decryption procedure to answer the query. Algorithm $\mathcal{S}$ ultimately forwards the guess of $\mathcal{A}$ to $\mathcal{C}$. As $\mathcal{S}$ perfectly simulates the LD-IND-CCA-game challenger to $\mathcal{A}$, its advantage in the IND-CCA game is at least that of $\mathcal{A}$ in the LD-IND-CCA game. If the latter were non-negligible, then so would be former, and the IND-CCA security of $\mathcal{E}_0$ would be contradicted. □

Proof (of Theorem 4 – Untraceability). As users encrypt their identities to request keysm if $\mathcal{E}_2$ is IND-CPA secure and PoK is ZK, the third-party cannot infer any information about the user identities during protocol KeyDer. Let $\mathcal{H}(1^\lambda)$ be an algorithm which interacts with an adversary $\mathcal{A}(1^\lambda)$ of the traceability experiment. It runs $\mathcal{E}.\mathsf{Setup}(1^\lambda)$, and sends the resulting public parameters $\mathcal{PP}$ to $\mathcal{A}$. When $\mathcal{A}$ sends a challenge tuple (esk_0, esk_1, t), algorithm $\mathcal{H}$ simply generates $id \in_R \mathcal{ID}$ and a pair of keys $(ek, dk) \leftarrow \mathcal{E}_2.\mathsf{Setup}(1^\lambda)$. Whenever $\mathcal{A}$ requests a key for a time period t, algorithm $\mathcal{H}$ computes $ct^{id} = \mathcal{E}_2.\mathsf{Enc}(ek, id)$, and sends

(ek, t, ct^{id}) to $\mathcal{A}$, simulates a proof for Relation 1 by running the simulator of PoK, and proceeds like $\mathsf{KeyDer.U}$ in the remaining of the protocol; and otherwise does not perform any computation. Since

$$\begin{aligned}
&\left|\Pr\left[\mathbf{Exp}^{\text{trace}-0}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{trace}-1}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right]\right| \leq \\
&\left|\Pr\left[\mathbf{Exp}^{\text{trace}-0}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{H}) = 1\right]\right| \\
&+ \left|\Pr\left[\mathcal{A}(\mathcal{H}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{trace}-1}_{\mathcal{E},\lambda}(\mathcal{A}) = 1\right]\right|,
\end{aligned}$$

if PoK is ZK and $\mathcal{E}_2$ is IND-CPA secure, then the upper bound is negligible, and the advantage of $\mathcal{A}$ is thus negligible. □

Proof (of Theorem 5 – Recoverability). It suffices to prove that with a probability close to δ, when given ciphertexts generated for the time period and the identity for which Rec is δ-correct, and encrypting messages with distribution $\mathcal{D}$, algorithm B requests the third-party secret key corresponding to the time period and the identity.

As soon as this event occurs, algorithm Rec runs extractor $\mathcal{K}$ to extract a secret. Since the commitment and the identity used in the witness for the proof are signed by the key-enhancement authority, algorithm Rec must send, with overwhelming probability, the commitment and an encryption of the identity that are in the user enhanced public key. As the commitment scheme is binding, the extracted secret is therefore the one that was used in the key-enhancement protocol.

To this end, consider the following hybrid algorithms that interact with a recoverability-game adversary $\mathcal{A}$, and each of which first proceeds like the recoverability-game challenger, ends up with a user enhanced public key that contains an identity id, and then receives from $\mathcal{A}$ an algorithm B claimed to be δ-correct in a time period t with respect to a distribution $\mathcal{D}$.

- $\mathcal{H}_0$ that generates $m \leftarrow_{\$} \mathcal{D}$, $m_1 \in_R \mathcal{M}$, submits ciphertext $(\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, m \oplus m_1), \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, m_1, \{t, id\}))$ to B. Whenever B sends a tuple $(ek', t', ct^{id'})$, and then succeeds in protocol PoK, algorithm $\mathcal{H}_0$ computes a derived secret-key $sk_{\mathcal{T}}^{t',id'} \leftarrow \mathsf{Der}_0(sk_{\mathcal{T}}, t', ek', ct^{id'})$, and sends it to B; and
- $\mathcal{H}_1$ which generates $m \leftarrow_{\$} \mathcal{D}, m_1, m_1' \in_R \mathcal{M}$, submits ciphertext $(\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, m \oplus m_1), \mathcal{E}_1.\mathsf{Enc}(pk_{\mathcal{T}}, m_1', \{t, id\}))$ to B, and then proceeds like $\mathcal{H}_0$.

Let E denote the event in which $id' = id$, and F the event in which B succeeds in protocol PoK. Assume $\Pr\left[\overline{F}\right]$ not to be nil, and observe that

$$\begin{aligned}
\Pr\left[\overline{E} \cup \overline{F}\right] &\leq \Pr\left[\overline{\mathsf{B}(\mathcal{H}_0) = m}\right] + \Pr\left[\left(\overline{E} \cup \overline{F}\right) \cap \{\mathsf{B}(\mathcal{H}_0) = m\}\right] \\
&\leq \Pr\left[\overline{\mathsf{B}(\mathcal{H}_0) = m}\right] + \Pr\left[\overline{E} \cap F\right] + \Pr\left[\overline{F} \cap \{\mathsf{B}(\mathcal{H}_0) = m\}\right] \\
&\leq \Pr\left[\overline{\mathsf{B}(\mathcal{H}_0) = m}\right] + \Pr\left[\overline{E} \cap F\right] + \Pr\left[\mathsf{B}(\mathcal{H}_0) = m | \overline{F}\right]. \qquad (2)
\end{aligned}$$

Lemma 1. *There exists an algorithm $\mathcal{S}$ such that*

$$\left|\Pr\left[\mathsf{B}(\mathcal{H}_0) = m|\overline{F}\right] - \Pr\left[\mathsf{B}(\mathcal{H}_1) = m|\overline{F}\right]\right| \leq \mathbf{Adv}^{\mathrm{ph-cpa}}_{\mathcal{E}_1,\lambda}(\mathcal{S}).$$

Proof. If algorithm B fails in protocol PoK, then it does not get a derived secret key from $\mathsf{KeyDer.T}$, and can only distinguish the two hybrids by distinguishing the plaintexts encrypted with $\mathcal{E}_1$. A complete proof is given below. □

Lemma 2. *There exists an algorithm $\mathcal{S}$ such that*

$$\Pr\left[\overline{E} \cap F\right] \leq \mathbf{Adv}^{\mathrm{euf-cma}}_{\mathcal{S},\lambda}(\mathcal{S}) + \mathsf{negl}(\lambda).$$

Proof. If algorithm B succeeds in protocol PoK without having sent an encryption of the identity that it was given, then by the soundness of the proof system, it must have forged a signature. A full proof is given below. □

In addition to that, $\Pr\left[\mathsf{B}(\mathcal{H}_1) = m|\overline{F}\right] \leq p(\mathcal{D})$ as the ciphertext that B receives bears no information about m_1. Algorithm B can therefore only guess m, and it can be done with probability at most $p(\mathcal{D})$, the predictive probablity of distribution $\mathcal{D}$. Therefore, combining the previous two lemmas, Equation 2 implies that

$$\begin{aligned}\Pr\left[\overline{E} \cup \overline{F}\right] &\lesssim \Pr\left[\overline{\mathsf{B}(\mathcal{H}_0) = m}\right] + \mathbf{Adv}^{\mathrm{ph-cpa}}_{\mathcal{E}_1,\lambda}(\mathcal{S}) + \mathbf{Adv}^{\mathrm{euf-cma}}_{\mathcal{S},\lambda}(\mathcal{S}) + p(\mathcal{D}) \\ &\lesssim 1 - \delta + \mathbf{Adv}^{\mathrm{euf-cma}}_{\mathcal{S},\lambda}(\mathcal{S}) + \mathbf{Adv}^{\mathrm{ph-cpa}}_{\mathcal{E}_1,\lambda}(\mathcal{S}) + p(\mathcal{D}).\end{aligned}$$

Thus, setting

$$\tilde{\delta} = \delta - \mathbf{Adv}^{\mathrm{euf-cma}}_{\mathcal{S},\lambda}(\mathcal{S}) - \mathbf{Adv}^{\mathrm{ph-cpa}}_{\mathcal{E}_1,\lambda}(\mathcal{S}) - \mathsf{negl}(\lambda) - p(\mathcal{D}) = \delta - p(\mathcal{D}) + \mathsf{negl}(\lambda),$$

it follows that $\Pr[E \cap F] \geq \tilde{\delta}$.

If $\Pr\left[\overline{F}\right]$ is nil, the conditional probability on $\overline{F}$ is not defined, but the lower-bound still holds.

Algorithm Rec then does the following. It repeatedly submits ciphertexts encrypting messages generated with distribution $\mathcal{D}$. If B engages in protocol PoK, algorithm Rec plays the role of the verifier. Until the end of the protocol, B cannot tell Rec and $\mathsf{KeyDer.T}$ apart by definition of the latter. For N such queries, B requests, with probability at least $1 - \left(1 - \tilde{\delta}\right)^N$, the secret key for the time period in which it is claimed to be δ-correct and the identity in the user enhanced public key. Performing N such queries for N large enough (e.g., $\omega(\log \lambda)$) makes it overwhelming. Once this event occurs, calling on $\mathcal{K}$, which can rewind B, a witness which contains a triple (c, s, o), with c the same as in the enhanced public key, can be recovered with overwhelming probability. Since $\mathcal{C}$ is binding, with overwhelming probability, s is the secret that was given by $\mathcal{A}$ during the key-enhancement protocol. □

Proof (of Lemma 1). Let $\mathcal{S}$ be an algorithm which interacts with $\mathcal{A}$, and attempts to distinguish two PH-CPA-game challengers: a challenger $\mathcal{C}_0$ that encrypts the

first message and a challenger $\mathcal{C}_1$ that encrypts the second. After receiving a public key $pk_{\mathcal{T}}$, algorithm $\mathcal{S}$ generates a commitment key ck, and sends $(ck, pk_{\mathcal{T}})$ to $\mathcal{A}$. When $\mathcal{A}$ sends a pair $(s, pk_{\mathcal{U}})$, algorithm $\mathcal{S}$ computes $(c, o) \leftarrow \mathsf{Com}(ck, s)$, sends (c, o) to $\mathcal{A}$, and executes the key-enhancement protocol with $\mathcal{A}$, running $\mathsf{KeyEn.A}(ck, c, pk_{\mathcal{U}})$ as a subroutine. $\mathcal{S}$ obtains an enhanced public key which contains an identity id. Whenever $\mathcal{A}$ requests the execution of $\mathsf{KeyDer.T}(sk_{\mathcal{T}}, ck)$ and sends a tuple $(ek', t', ct^{id'})$, if $t' > t_c$, then $\mathcal{S}$ returnrs $\perp$, otherwise $\mathcal{A}$ and $\mathcal{S}$ engage in protocol PoK as prover and verifier respectively. If $\mathcal{A}$ succeeds in protocol PoK, algorithm $\mathcal{S}$ runs $\mathcal{K}$, gets a witness that contains an identity id' with overwhelming probablility, and queries the secret key $sk_{\mathcal{T}}^{t',id'}$ for attribute set $\{t', id'\}$ to the challenger with which it interacts. Algorithm $\mathcal{S}$ then returns $\mathsf{Der}_2(sk_{\mathcal{T}}^{t',id'}, ek')$ to $\mathcal{A}$. If $\mathcal{A}$ does not succeed in protocol PoK, algorithm $\mathcal{S}$ sends $\perp$ to $\mathcal{A}$. When $\mathcal{A}$ sends an algorithm B, a distribution $\mathcal{D}$ and a time period $t > t_c$, algorithm $\mathcal{S}$ generates $m \leftarrow_{\$} \mathcal{D}$, $m_1, m_1' \in_R \mathcal{M}$, sends $(m_1, m_1', \{t, id\})$ to the challenger with which it interacts, and gets back a challenge ciphertext ct^*. It then submits $(\mathcal{E}_0.\mathsf{Enc}(pk_{\mathcal{U}}, m \oplus m_1), ct^*)$ to B. When B sends a tuple $(ek', t', ct^{id'})$ (if it does at all), conditioned on event $\overline{F}$ (i.e., B does not succeed in protocol PoK), algorithm $\mathcal{S}$ returns $\perp$. Algorithm $\mathcal{S}$ eventually outputs 1 if B outputs m, and otherwise (B outputs $m' \neq m$ or $\perp$) outputs 0. Conditioned on $\overline{F}$, in case the challenger is $\mathcal{C}_0$, algorithm $\mathcal{S}$ perfectly simulates $\mathcal{H}_0$ to $\mathcal{A}$. Analogously, if the challenger is $\mathcal{C}_1$, algorithm $\mathcal{S}$ perfectly simulates $\mathcal{H}_1$ to B, hence the claim. □

Proof (of Lemma 2). Let $\mathcal{S}$ be an algorithm which interacts with $\mathcal{A}$ and the existential-forgeability-game challenger $\mathcal{C}$. Upon receiving a verification key vk, algorithm $\mathcal{S}$ generates a commitment key ck and third-party pair of keys $(pk_{\mathcal{T}}, sk_{\mathcal{T}})$, and sends $(ck, pk_{\mathcal{T}})$ to $\mathcal{A}$. Algorithm $\mathcal{S}$ then proceeds like the recoverablity-game challenger until the key-enhancement protocol, in which, instead of generating a signature pair of keys, asks $\mathcal{C}$ to sign the commitment and the identity (denote it id) involved in the protocol. It carries on as the simulator of Lemma-1 until B sends a tuple $(ek', t', ct^{id'})$ for $id' \neq id$, and succeeds in protocol PoK as prover, which occurs in event $\overline{E} \cap F$. Algorithm $\mathcal{S}$ runs $\mathcal{K}$ to extract, with overwhelming probability, a witness which contains a commitment c, identity id' and a signature σ such that $\mathsf{Verify}(vk, (c, id'), \sigma) = 1$. Algorithm $\mathcal{S}$ then sends $((c, id'), \sigma)$, as a forgery, to $\mathcal{C}$. As $\mathcal{S}$ perfectly simulates the recoverability-game challenger to $\mathcal{A}$ conditioned on $\overline{E} \cap F$, the claim follows. □

Multi-User Case. Should there be several users, the key-enhancement protocol is executed once for each of the users with the restriction that a user's identity is chosen uniformly at random from the set of non-yet attributed identities. The correctness, privacy, LD-CCA security and untraceability are proved in the very same manner as in the single-user case. To recover the secret of at least one of a set of colluding users, the recovery algorithm chooses uniformly at random the enhanced public key of one of