Traitor Deterring Schemes: Using Bitcoin as Collateral for Digital Content

Aggelos Kiayias
National and Kapodistrian University of Athens
Dept. of Informatics and Telecommunications
aggelos@di.uoa.gr

Qiang Tang
University of Connecticut
Dept. of Computer Science and Engineering
qtang84@gmail.com

ABSTRACT

We put forth a new cryptographic primitive called a *Traitor Deterring Scheme* (TDS). A TDS is a multi-recipient public-key encryption scheme where an authority issues decryption keys to a set of users. The distinguishing feature of a TDS is that secret-keys are issued only after the users provide some private information as a form of *collateral*. The traitor deterring property ensures that if a malicious coalition of users (aka "traitors") produces an unauthorized (aka "pirate") decryption device, any recipient of the device will be able to recover at least one of the traitors' collaterals with only black-box access to the device. On the other hand, honest users' collaterals are guaranteed to remain hidden. In this fashion a TDS deincentivizes malicious behavior among users.

We model, construct and analyze TDS's based on various cryptographic assumptions and we show how bitcoin can be used as collateral for real world deployment of TDS's for the distribution of digital content. Along the way, we present cryptographic building blocks that may be of independent interest, namely fuzzy lockers, and comparison predicate encryption schemes for exponentially large domains. We also compare TDS with previous primitives specifically traitor tracing schemes (TTS) introduced by Chor et al. [9] and digital signets for self enforcement introduced by Dwork et al. [12]. A TDS constitutes a strict strengthening of a TTS and, when modeled in what we call the "known ciphertext model", it is a reformulation of digital signets in the public-key, black-box secure setting. In digital signets the adversary attempts to transmit a pirate copy at a favorable "space rate", i.e., without having to send the whole plaintext (and without revealing the traitor collaterals). It is an open question from [12] to construct $o(1)$ space rate schemes under a falsifiable assumption. With our TDS constructions we resolve this open question showing feasibility for space rates $O(\log\lambda/\lambda)$ and infeasibility for space rates $\Omega(\log^2\lambda/\lambda)$.

CCS'15, October 12–16, 2015, Denver, Colorado, USA.
© 2015 ACM. ISBN 978-1-4503-3832-5/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810103.2813698.

Categories and Subject Descriptors

K.6 [**Management of Computing and Information Systems**]: Security and Protection; E.3 [**Data Encryption**]: Public key Cryptosystems

Keywords

Digital Rights Management; Public-key Cryptography; Self-enforcement; Key Management; Bitcoin

1. INTRODUCTION

A traitor tracing scheme (TTS) is a multi-user encryption scheme that when some users (aka traitors) collude to produce an unauthorized decryption device (aka a pirate decryption box), it is possible to recover at least one of their identities. TTS's de-incentivize piracy, in the sense that colluders may be identified by the authority once an unauthorized device is detected. Since it was introduced in [9], there have been numerous works, improving different aspects of efficiency and security considerations, cf. [2–5, 8, 22, 23]. However, in a TTS, recovering the identity of a traitor can only happen when the authority becomes aware of the unauthorized decryption device. This means that if the traitors operate stealthily (e.g., distribute a pirate device in some closed network) there is nothing the authority can do to deter them, and thus in this setting the tracing mechanism becomes ineffective. Furthermore, the penalty that the authority may inflict to the traitors can only be applied "after-the-fact", i.e., only after the unauthorized decoder has been recovered and analyzed by the authority.

To address the challenges above, we strengthen the notion of TTS and put forth a new primitive we call a *traitor deterring scheme* (TDS): a multi-recipient encryption scheme where each recipient (henceforth also called a user) has some secret information that is provided as a collateral and hidden in a public directory. If the user is honest and keeps her key to herself, her secret information remains hidden. On the other hand, if some users collude to produce an unauthorized decryption device, any recipient of the device will be able to recover one of the colluders' collateral secret information.

One particularly suitable user-specific information that can be used as collateral within a TDS is a bitcoin address secret key. When registering for service, the subscriber puts as collateral a small bitcoin amount into a fresh address and the secret-key of the bitcoin address is embedded as collateral. In case the bitcoin address is used as input to a transaction, the public nature of the bitcoin ledger enables the service provider to detect it and take appropriate action (See section 6 for details).

Compared to TTS's, the main difficulty of constructing a TDS is that one needs to enable a public recovering procedure which returns the user's secret information that is an element of an exponentially sized domain — in other words linear number of bits in the security parameter λ need to be extracted from the pirate box. Contrary to that, in a TTS, the authority only needs to recover the identity of a traitor, which is an element of merely a polynomially sized domain — in other words just logarithmic number of bits in the security parameter λ need to be extracted from the pirate box. As in TTS, the recovering procedure should work given only black-box access to the pirate decryption box which may be only partially working. Furthermore, it should operate without utilizing any private-key information, as in a TTS with public traceability [8].

A TDS (or a TTS) can also be considered in a stronger adversarial model that we call "the known ciphertext model". In this model the adversary aims at communicating a pirated copy consisting of a sequence of plaintexts that corresponds to a given set of (polynomially many) ciphertexts (e.g., the contents in a CD or a public database); without loss of generality we can assume the pirate copy is in the form of a pirate box that acts only on the known sequence of ciphertexts. The adversary aims at producing a pirate box of smaller size than the sequence of plaintexts; we capture this in the model by requiring the "space rate" of the attacker to be $o(1)$. This problem was first considered by Dwork, Lotspiech and Naor [12]. Constructing a TDS or a TTS in the known ciphertext model under a falsifiable assumption has been an open question since then.

Our contributions. We formalize TDS's and we give two different construction methods that we instantiate in various ways; further, we formalize the known-ciphertext model in the spirit of [12] and we provide both feasibility and infeasibility results for TDS in this model. Finally we elaborate on how to use bitcoin as collateral in conjunction to a TDS. In more detail:

1. We put forth the formal model for TDS's. Such schemes enable the embedding of hidden user-specific information in a public parameter, and have three security properties: (i) security of plaintexts which is formalized in a standard fashion as in public-key encryption; (ii) privacy of user information that is hidden in the public parameters. This property should be upheld even if all other users conspire against a user as long as the secret key of the user is not compromised; finally, (iii) traitor deterring suggests that there is a recoverability algorithm that given black-box access to some working implementation of a decryption device, it is capable of recovering the private information of at least one traitor, using only public information.
2. We give two construction methods for TDS's. The first one is based on fingerprinting codes [9, 20] and a new primitive we call a fuzzy locker. In a fuzzy locker, the message is encrypted using a random codeword C_i; the decryption operation returns the message given any C^* that would result in the i-th user being accused in the underlying fingerprinting code. In the TDS construction, the recovering procedure will first retrieve a pirate codeword C^* from the decryption device; the traceability of the fingerprinting code will guarantee that one of the collusion's codewords will be accused, thus the corresponding traitor secret will be unlocked. We then give a concrete construction of a fuzzy locker for CFN codes [9] using the idea of fuzzy extractors [11] paired with efficient list decoding for Reed-Solomon codes [17,30]. Our second construction method for TDS's generalizes the constructions of [4, 5] that are based on comparison predicate encryption (CPE). Contrary to these works however, we require that the user identity space is exponentially large, so that a randomly chosen user identity can be used as a secret key to hide the user secret information directly. To recover the hidden information given a pirate decryption decoder we utilize a binary search type of recovering mechanism to navigate through the exponentially sized domain and discover one of the traitor identities. Given this identity we proceed to unlock the user hidden data. A CPE scheme can be obtained via functional encryption (FE) using indistinguishability Obfuscation (iO) [13]. In order to obtain a construction based on standard assumptions we resort to bounded collusion FE [14, 15]. We provide a more efficient construction for this primitive via a combinatorial argument and we then use it to instantiate a CPE with exponential size domain. Our TDS constructions are summarized in Fig. 1.
3. We revisit the problem of digital signets [12] and we formulate the "known ciphertext model" for TDS where the adversary knows the target set of (polynomially many) ciphertexts before implementing the pirate box. In an attack in this model, the adversary tries to achieve a favorable "space rate", i.e., produce a decryption box that has size smaller than the total plaintext material that is encoded in the known ciphertexts without leaking any of the traitors' collaterals. Constructing a TDS in the known ciphertext model is equivalent to the problem of constructing digital-signets with self-enforcement as defined in [12] which is open under falsifiable assumptions; the construction of [12] assumes an *incompressible function* of a specific type (this is an unfalsifiable assumption) and the recovering strategy has full access to the key. It works for any space rate $o(1)$. With our TDS constructions we resolve the open question showing feasibility under falsifiable assumptions for space rates $O(\log \lambda/\lambda)$ while we show infeasibility for space rates $\Omega(\log^2 \lambda/\lambda)$ in the black-box recoverability setting. In our results, we exploit bounds on the false positive rate of the membership testing problem to show how our TDS schemes can be used while our negative result applies Bloom filters [1] to provide an efficient attacker strategy.
4. We describe how one can use bitcoin as a collateral in a TDS. Recall that collaterals are arbitrary strings hence a service provider (SP) can embed as collateral the secret-key of a fresh bitcoin address credited by the user. As part of the user agreement, the account should remain frozen (i.e., any outgoing transaction from this account can be noticed by the service provider from the public ledger and the subscription will be cancelled). As long as the user respects the service agreement the collateral remains safe and the user may reclaim it when the service contract terminates.

	Assumption	Ciphertext size	Upper bound on t	Recoverability
Construction I	PKE	$O(t^2 \log^2(n/\epsilon))$	$O(\log(n/\epsilon)/\lambda)$	Black-box
Construction I	PKE	$O(t^4\lambda)$	n	Black-box
Construction II	LWE	$O(t^{3+e}\text{poly}(\lambda))$	n	Black-box
Construction II	iO	$O(1)$	n	Black-box

Figure 1: Comparison of our TDS's; t is the collusion size, n is total number of users, $e = 1/\mathbf{poly}(\lambda)$, ϵ is the error term in the fingerprinting code which is $\mathsf{negl}(\lambda)$ and λ is the security parameter. PKE denotes public-key encryption, LWE denotes the learning with errors problem, and iO denotes indistinguishability obfuscation.

Related primitives. As discussed above, a TTS aims at providing "a posteriori" deterrence of malicious users while TDS provides, in addition, a proactive way of deterrence. Furthermore, traitor tracing is possible only when the authority gains access to the pirate box, while in a TDS, the mere act of sharing a decryption key (or any, even partially working, implementation of a decryption algorithm containing such key) will lead to the leakage of one of the traitors' secrets. We show that a traitor deterring scheme implies a publicly traceable traitor tracing scheme [8] (cf. Section 2).

Another closely related notion to a TDS is digital signets for self-enforcement [12]. In this multi-user encryption system, the adversary that controls a set of traitor user keys and wants to retransmit a certain plaintext that was transmitted, will either send a message as long as the plaintext itself, or will have to leak some of the traitors' private data. The formalization of the problem in [12] assumes that recoverability of the collateral information requires direct access to the traitor keys (also called white-box access). In our terminology, they provide a symmetric-key TDS that is only secure in the non-black-box sense. The construction provided by [12] relies on the unfalsifiable assumption that $f(x) = g_1^x || g_2^x || \dots || g_\ell^x$ is incompressible (incompressible means given x, f, no adversary can come up with an intermediate value y, such that: (1). $|y|/|f(x)| = o(1)$; (2). one can recover $f(x)$; (3). x remains hidden).

Kiayias and Tang studied the problem of leakage deterring (LD) public key cryptography [21]. If a key owner leaks any partially working decryption box, a piece of secret information that is embedded in her public key will be revealed to the recipient. Our notion of TDS is a generalization of LD from the single user setting to the multi-user setting. We note that because of collusion attacks in the multi-user setting the techniques for recoverability from [21] are not directly applicable for building a TDS (even a scheme with ciphertext size linear in the number of users is not obvious).

2. DEFINITIONS AND SECURITY MODELS

We provide the formal definition and security model of TDS's and demonstrate their relationship to TTS's.

Syntax of traitor deterring schemes. Informally, a traitor deterring scheme is a multi-user encryption scheme with a deterring mechanism such that if some of the receivers collude to leak an implementation of a (potentially only partially working) decryption device, any recipient of the device will be able to recover one of the colluding user's secret information which is hidden in the public parameter of the system. Formally we have the following:

- **Setup**$(1^\lambda, s_1, \dots, s_n)$: This algorithm is composed of two parts: **KeyGen**, which, on input the security parameter it outputs an encryption key pk, and a set of decryption keys $sk_1, \dots, sk_n$; and **ParGen** that on input $pk, sk_1, \dots, sk_n$ and the users' secrets $s_1, \dots, s_n \in \{0,1\}^\lambda$ it outputs public parameter $para$.
- **Enc**(pk, m): on input $para, pk$ and a message m, it outputs a ciphertext c.
- **Dec**(sk_i, c): on input $para, pk$, one of the secret keys sk_i and a ciphertext c, it outputs a plaintext m.
- **Rec**$^{B,\mathcal{D}}(pk, para)$: on input $para, pk$ with has oracle access to a device B and a distribution $\mathcal{D}$ it outputs a string in $\{0,1\}^\lambda$ or $\perp$.

The correctness of the scheme is standard and entails that **Enc**$(pk, \cdot)$ can be inverted by **Dec**$(sk_i, \cdot)$ for any $i = 1, \dots, n$. The intended functionality of the algorithm **Rec** is that if B is created by a collusion of receivers with secret keys $sk_{i_1}, \dots, sk_{i_t}$ and operates correctly for ciphertexts whose corresponding plaintext follows a distribution $\mathcal{D}$, the algorithm outputs at least one of the strings $s_{i_1}, \dots, s_{i_t}$. We clarify the conditions under which this is supposed to happen (as well as the other necessary security properties) in the next paragraph.

Security model. There are three security requirements for a traitor deterring scheme, *security* of plaintexts, *privacy* of user's secrets, and *traitor deterring.*

IND-CPA security. Regarding security of plaintexts we consider a security property of IND-CPA defined in a standard fashion: the challenger $\mathcal{C}$ runs setup to obtain $s_1, \dots, s_n$ and provides the adversary $\mathcal{A}$ with $para, pk$. In response, $\mathcal{A}$ provides two plaintexts m_0, m_1 to $\mathcal{C}$. Subsequently $\mathcal{C}$ computes $\psi = \mathbf{Enc}(pk, m_b)$ for a random $b \in \{0,1\}$ and provides ψ to $\mathcal{A}$. $\mathcal{A}$ returns a bit b' and $\mathcal{C}$ terminates with 1 if $b = b'$ and 0 otherwise. The probability that $\mathcal{C}$ returns 1 means that $\mathcal{A}$ is successful and we denote it by $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{indcpa}}(1^\lambda)$. For security to hold it must be that $\Pr[\mathsf{Succ}_{\mathcal{A}}^{\mathsf{indcpa}}(1^\lambda)] \leq 1/2 + \mathsf{negl}(\lambda)$. The notion of security can be extended in a straightforward manner to IND-CCA2.

Privacy. Regarding the privacy of user secret information it should be the case that each s_i value remains hidden within the public parameter even all other users are corrupted. Formally, consider the following game:

- The challenger $\mathcal{C}$ first simulates the **KeyGen** part of the **Setup** algorithm and returns pk to the adversary.
- The adversary $\mathcal{A}$ sends an index i as well as private information $\{s_j\}_{j \neq i}$ and the pair $s_{i,0}, s_{i,1}$ to $\mathcal{C}$.
- The challenger $\mathcal{C}$ randomly flips a bit b, and simulates the **ParGen** part of **Setup** on input $pk, sk_1, \dots, sk_n$, $s_1, \dots, s_{i-1}, s_{i,b}, s_{i+1}, \dots, s_n$. $\mathcal{C}$ sends to $\mathcal{A}$ the values $para, sk_1, \dots, sk_{i-1}, sk_{i+1}, \dots, sk_n$.
- $\mathcal{A}$ returns a single bit b' and $\mathcal{C}$ returns 1 if $b = b'$.

The event that $\mathcal{C}$ returns 1 means $\mathcal{A}$ is successful and we denote it by $\mathsf{Succ}_{\mathcal{A}}^{\mathtt{priv}}(1^\lambda)$. For privacy of secret information to hold it must be that $\Pr[\mathsf{Succ}_{\mathcal{A}}^{\mathtt{priv}}(1^\lambda)] \leq 1/2 + \mathsf{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$. Note that letting the challenger send the public key first makes the definition stronger. It is also possible to define weaker variants of the above definition, e.g., where $\mathcal{A}$ is restricted to a number t of secret-keys or the pk is returned together with the secret keys.

Traitor-Deterring. Finally we define the traitor deterring property. In order to specify the definition we need first to define the notion of δ-correctness with respect to a public-key pk and a plaintext distribution $\mathcal{D}$. A device B is $\delta-$correct with respect to $\mathcal{D}$ and pk if it satisfies that $\Pr[B(\mathbf{Enc}(pk, m)) = m : m \leftarrow \mathcal{D}] \geq \delta$. With the public parameter, and a non-trivial pirate decryption box B which is created by the collusion of all users, the recovering algorithm should determine of the colluder's secret information s_i. Formally, consider the following game:

- The challenger $\mathcal{C}$ simulates the **Setup** algorithm and the adversary $\mathcal{A}$ receives pk. $\mathcal{A}$ then provides a vector of secret information $s_1, \ldots, s_n$ as well as an arbitrary subset $T \subseteq \{1, \ldots, n\}$ to the challenger $\mathcal{C}$ and $\mathcal{A}$ receives the secret keys of all users in T, $\{sk_i \mid i \in T\}$ as well as the public parameter $para$.
- $\mathcal{A}$ outputs an implementation B and a distribution $\mathcal{D}$.
- $\mathcal{C}$ returns 1 iff $\mathbf{Rec}^{B,\mathcal{D}}(pk, para) \notin \{s_i \mid i \in T\}$.

We define by $\mathsf{Succ}_{\mathcal{A}}^{\mathtt{deter}}(1^\lambda)$ the event that $\mathcal{C}$ returns 1. We say a scheme achieves fully collusion resilient, black-box traitor deterring w.r.t. a class of distributions $\mathfrak{D}$ (that may depend on δ) if for any PPT adversary $\mathcal{A}$ it holds that

$$\Pr[\text{B is } \delta\text{-correct w.r.t. } \mathcal{D} \wedge \mathcal{D} \in \mathfrak{D} \wedge \mathsf{Succ}_{\mathcal{A}}^{\mathtt{deter}}(1^\lambda)] = \mathsf{negl}(\lambda).$$

In the above experiment we assume that **Rec** has resettable black-box access to B. Weaker variants of the above formulation may be relevant in some settings and can be "t-collusion resilient" (as opposed to fully-collusion resilient) or they may extend **Rec**'s access to B (e.g., in a non-black-box setting **Rec** may have access to the traitor keys).

Definition 2.1 $\langle$**Setup**, **Enc**, **Dec**, **Rec**$\rangle$ *is a (fully-collusion resistant, black-box) traitor deterring scheme if it satisfies, (i) correctness, (ii) IND-CPA security, (iii) privacy and (iv) fully-collision resistant, black-box traitor deterring.*

TDS and TTS. We conclude the section by a brief argument that a traitor deterring scheme is a strict generalization of a traitor tracing scheme (in fact of a TTS with "public-traceability" [8]). Given a TDS: $\langle$**Setup**, **Enc**, **Dec**, **Rec**$\rangle$, the reduction is easy with the following simple observation. First we set $s_i = i$ for all $i = 1, \ldots, n$. It follows that the **Setup** algorithm requires no other input other than the security parameter λ. Observe now that the **Rec** algorithm will output one of the indices of the colluding users who jointly produce the decryption box B with only access to pk, hence it is a TTS with public-traceability.

3. TRAITOR DETERRING FROM FINGERPRINTING CODES

In this section, we will present our first technique of constructing a TDS from fingerprinting codes. We first formalize a new encryption scheme we call fuzzy locker (w.r.t a fingerprinting scheme), from which together with a public key encryption, we will construct a TDS. We then give a concrete construction of fuzzy locker for CFN codes [9].

First, let us recall the definition of fingerprinting codes [20]. A *q-ary fingerprinting code* is a pair of algorithms $(\mathsf{Gen}, \mathsf{Accuse})$. Gen is a probabilistic algorithm with input a security/error parameter ϵ and two numbers n, t denoting the number of users and the maximum collusion size respectively, and $t \in [n] = \{1, \ldots, n\}$. It outputs n q-ary strings $\mathcal{C} = \{C_1, \ldots, C_n\}$ (called codewords), where $C_i = \mathsf{c}_1^i \ldots \mathsf{c}_\ell^i$ for $i \in [n], j \in [\ell], \mathsf{c}_j^i \in Q$–the alphabet set with size q and a tracing key tk. Accuse is a deterministic algorithm with input a "pirate" codeword C^*, and a user codeword C_i and the tracing key tk; it outputs a bit in $\{0, 1\}$.

Suppose adversary $\mathcal{A}$ corrupts up to t users (whose indices form a set $\mathcal{U}_{cor} \subset [n]$), and outputs a pirate codeword $C^* = \mathsf{c}_1^* \ldots \mathsf{c}_\ell^*$. We define the accused user set as $\mathcal{U}_{acc} = \{i \in [n] : \mathsf{Accuse}(tk, C^*, C_i) = 1]$. A fingerprinting code is called *$t-$collusion resistant* (*fully collusion resistant* if $t = n$) if it satisfies: (i) *traceability*, if the strategy of producing C^* satisfies the "marking assumption", (for each $i \in [n], \mathsf{c}_i^* = \mathsf{c}_i^j$ for some $j \in \mathcal{U}_{cor}$), then one of the colluders must be accused, i.e., $\Pr[\mathcal{U}_{acc} \cap \mathcal{U}_{cor} = \emptyset] \leq \epsilon$; and (ii) *soundness*, the probability that an innocent user is accused is bounded by ϵ, i.e., $\Pr[([n] - \mathcal{U}_{cor}) \cap \mathcal{U}_{acc} \neq \emptyset] \leq \epsilon$.

3.1 TDS from fuzzy lockers.

Fingerprinting codes are combinatorial designs that enable testing whether a codeword is used in generating a pirate codeword. They were demonstrated to be very useful in building TTS in a line of previous works, e.g., [3,9,22]. The basic idea is that each user will be assigned an "identity" which is represented by a codeword, and the secret keys for the user will be selected from a group of keys according to his codeword. The encryption algorithm will cover all the user keys. The tracing algorithm will first recover a "pirate codeword" by feeding the pirate decryption device with malformed (but seemingly valid in the view of $\mathcal{A}$) ciphertexts, and then it will run the tracing algorithm of the fingerprinting code to identify at least one of the colluding users who participated in producing the pirate codeword.

The main challenge of upgrading the above paradigm to a TDS is the way of embedding and recovering of the secret information of the users. To address this, we formalize a new primitive we call fuzzy locker w.r.t. a (publicly traceable) fingerprinting code. In a fuzzy locker, a message is encrypted using a random codeword C_i. The message can be decrypted ("unlocked") only if one provides a pirate codeword C^* such that C_i will be accused by the accusation algorithm, otherwise, the message will remain IND-CPA secure. Given such a primitive, one can construct a TDS as follows: the embedding of the user private information can be simply done via encryption using the user's codeword (which is normally randomly selected according to the Gen algorithm). The *privacy* requirement can be easily achieved via the security of the fuzzy locker. The recover algorithm will first retrieve a "pirate codeword" from the pirate box and then it will try decrypting all locked data using this pirate codeword. The *traitor deterring* property can be guaranteed by the traitor tracing property of the fingerprinting code, since at least one of the codewords used in producing the pirate codeword will be accused and thus the private user data can be retrieved.

We first give the formal definition and security model of a fuzzy locker. W.l.o.g., we can think of the Gen algorithm of the fingerprinting code $\mathcal{C}$ to operate in two phases, first, using n, t and the security parameter produces a secret state st and then uses a $\mathcal{C}.\mathsf{Sample}$ subroutine that produces the codewords one-by-one while updating the state st.

Definition 3.1 *A fuzzy locker w.r.t a (publicly traceable) fingerprinting code $\mathcal{C}$ consists of the following two algorithms:*

- $\mathbf{FL.Enc}(C_i, m)$*: Given a codeword $C_i \leftarrow \mathcal{C}.\mathsf{Sample}$ and a message m, the encryption algorithm outputs a ciphertext c.*
- $\mathbf{FL.Dec}(C^*, c)$*: Given a ciphertext c and a string C^*, the algorithm outputs a message m or $\perp$.*

Correctness: If $\mathcal{C}.\mathsf{Accuse}(tk, C_i, C^*) = 1$:

$$\Pr[\mathbf{FL.Dec}(C^*, c) = m] \geq 1 - \mathsf{negl}(\lambda).$$

Security of a fuzzy locker. We define t-resilient security (fully resilient if $t = n$) of a fuzzy locker scheme in the sense of IND-CPA security, by considering the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- The challenger produces st using Gen on input ϵ, t, n and sends $C_1, \ldots, C_t \leftarrow \mathcal{C}.\mathsf{Sample}(st)$ to $\mathcal{A}$.
- $\mathcal{A}$ selects two messages m_0, m_1 and sends them to $\mathcal{C}$.
- The challenger randomly samples a codeword $C_0 \leftarrow \mathcal{C}.\mathsf{Sample}(st)$, randomly flips a coin b, and sends $c = \mathbf{FL.Enc}(C_0, m_b)$ to the adversary $\mathcal{A}$.
- $\mathcal{A}$ outputs her guess b'.

A fuzzy locker is t-resilient IND-CPA secure if:

$$\mathsf{Adv}_{\mathsf{FL}}^{\mathcal{A}} = |\Pr[b' = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

Construction-I of TDS. Given a fuzzy locker and a public key encryption (PKE) with the algorithms (KeyGen, Enc, Dec), we can construct a TDS as follows:

- $\mathbf{Setup}(1^\lambda, s_1, \ldots, s_n, t)$: The algorithm first runs the (q-ary) codeword generation algorithm $\mathcal{C}.\mathsf{Gen}$ which inputs the security parameter, and t, n, it returns $\{C_i\}_{i \in [n]}$, tk, where $C_i = \mathsf{c}_1^i, \ldots, \mathsf{c}_\ell^i$. The algorithm then runs the KeyGen of the PKE and returns $q\ell$ key pairs:

$$\begin{array}{c}(pk_{1,1}, sk_{1,1}), \ldots, (pk_{1,\ell}, sk_{1,\ell});\\ (pk_{2,1}, sk_{2,1}), \ldots, (pk_{2,\ell}, sk_{2,\ell});\\ \cdots\\ (pk_{q,1}, sk_{q,1}), \ldots, (pk_{q,\ell}, sk_{q,\ell})\end{array}$$

  Finally, the **Setup** algorithm takes users' secrets $s_1, \ldots, s_n$, $tk, C_1, \ldots, C_n$ and all those key pairs as inputs, and it outputs system parameter $para$, an encryption key pk, and a set of decryption keys $sk_1, \ldots, sk_n$. Specifically, pk contains all the public keys above; $sk_i = \{sk_{1,\mathsf{c}_1^i}, \ldots, sk_{\ell,\mathsf{c}_\ell^i}\}$ for $i \in [n]$; and $para$ contains tk and $\langle \omega_1, \ldots, \omega_n \rangle$, where $\omega_i = \mathbf{FL.Enc}(C_i, s_i)$.
- $\mathbf{Enc}(pk, m)$: This algorithm is given pk and a message m. It first randomly samples $m_1, \ldots, m_{\ell-1}$, then computes $m_\ell = m - \sum_{i=1}^{\ell-1} m_i$ and $ct_{i,j} = \mathrm{Enc}(pk_{i,j}, m_j)$ for $i \in [q], j \in [\ell]$; it outputs the ciphertext $ct = \{ct_{i,j}\}$.
- $\mathbf{Dec}(sk_i, ct)$: This algorithm takes inputs $para, pk$, a secret key sk_i and a ciphertext ct. It parses the secret key and the ciphertext, and computes $m_j = \mathrm{Dec}(sk_{j,\mathsf{c}_j^i}, ct_{i,j})$ and further $m = \sum_{i=1}^{\ell} m_i$. The algorithm outputs m.
- $\mathbf{Rec}^{B,\mathcal{D}}(pk, para)$: This algorithm inputs $para, pk$ and has oracle access to a device B and a distribution $\mathcal{D}$. It first runs the following procedure for each index $k \in [\ell]$ to extract a pirate codeword from B:
  1. It initializes the pointer value $i_0 = 1$.
  2. It samples $m \leftarrow \mathcal{D}$, samples messages $\{m_i\}_{i \neq k}$ and computes $m_k = m - \sum_{i \neq k} m_i$.
  3. It feeds ciphertext $\{ct_{i,j}\}$ to B where $c_{i,j} = \mathrm{Enc}(pk_{i,j}, m_j)$ if $j \neq k$ or $i > i_0$; and $c_{i,k} = \mathrm{Enc}(pk_{i,k}, r_i)$ for $i \leq i_0$ where r_i is a random message.
  4. If in N runs (a value that will be determined in the analysis), the number of times n_1 that B returns m correctly is sufficiently smaller than that in the $(i_0 - 1)-$th experiment (the difference is denoted by n_0.), or it returns $r_{i_0} + \sum_{j \neq k} m_j$ the algorithm returns $\mathsf{c}_k^* = i_0$; otherwise, it stores n_1, sets $i_0 = i_0 + 1$, and repeats from step 2.

  The pirate codeword retrieved is C^*. The algorithm parses $para$ to identify the data $\langle \omega_1, \ldots, \omega_n \rangle$. It then runs the decryption algorithm of the fuzzy locker on all of them, i.e, for $i \in [n]$, it runs $\mathbf{FL.Dec}(C^*, \omega_i) = s_i^*$. The algorithm stops if $\exists s_i^* \neq \perp$ and it returns s_i^*.

Security analysis. Due to lack of space, we present here only a brief sketch about the security properties, and refer to the full version for the detailed proofs.

Regarding *IND-CPA security*, it follows in a straightforward manner from the security of the underlying PKE scheme.

Regarding *privacy* of the honest user secrets, follows easily from the security of the fuzzy locker.

Regarding the *black-box traitor deterring* property, note that the **Rec** algorithm proceeds in two steps, it first recovers a pirate codeword C^* from the box B. If there exists a colluder codeword C_i, s.t., $\mathsf{Accuse}(tk, C^*, C_i) = 1$, then in the second step, according to the correctness of the fuzzy locker, the decryption of $\mathbf{FL.Dec}(C^*, \omega_i)$ will return s_i. The security of the fingerprinting code guarantees that if any pirate codeword is produced following the "marking assumption", it can be used to accuse at least one of the colluders. The IND-CPA security of the underlying PKE scheme essentially enforces the "marking assumption." To see this, suppose the collusion user secret keys are $\{sk_i\}$ for $i \in \mathcal{U}_{cor}$, for each index j, the alphabet c_i^* for the pirate codeword at index i can not be any c_i^k for $k \notin \mathcal{U}_{cor}$ with probability significantly larger than the guessing probability $\delta - \alpha$. Otherwise, these keys may be used to decrypt a ciphertext encrypted under a public key $pk_{k,i}$.

The choice of N, n_0 can be easily determined as follows. There must exist an index i_0 such that the probability of returning a correct plaintext (denoted by p_1) is at least $[\delta - (\delta - \alpha)]/q = \alpha/q$ smaller than that for $i_0 - 1$ (denoted by p_2). From the Chernoff bound $\Pr[X < (1-\omega)\mu] \leq e^{-\omega^2\mu/2}$, let us use $X_i^1 = 1$ denote the event that decryption query for $i_0 - 1$ is answered correctly while $X_i^2 = 0$ denote that for i_0 is not answered correctly. Also we use $X_i = 1$ denote the above joint event, i.e., $\Pr[X_i = 1] = \Pr[X_i^1 = 1 \wedge X_i^2 = 0] = p_1(1-p_2) \geq p_1 - p_2 \geq \alpha/q$. When we set $N = \frac{q}{\alpha}\log^2\lambda, n_0 = \frac{\alpha}{2q}N$, where λ is the security parameter, the gap will almost always appear and thus a pirate codeword will be identified.

Theorem 3.2 *Given a public key encryption scheme, and a fully secure fuzzy locker (for a q-ary fingerprinting code), there exists a TDS satisfying: fully collusion resilient, black-*

box traitor deterring property w.r.t to any message distribution $\mathcal{D}$ that has min-entropy $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta - \alpha)$, where δ is the correctness required by the adversarial device, α is a non-negligible amount that is significantly smaller than δ, and the parameters are set to be $N = \frac{q}{\alpha}\log^2\lambda, n_0 = \frac{\alpha}{2q}N$.

3.2 A Fuzzy locker for CFN codes.

We now propose a construction for a fuzzy locker w.r.t. CFN codes [9].Consider the CFN fingerprinting scheme where the collusion size is set to be t; in order generate a codeword for a user j, the authority randomly samples $\mathsf{c}_1^j, \ldots, \mathsf{c}_\ell^j \leftarrow [q]^\ell$. The tracing algorithm accuses the user whose codeword has the largest number of locations that share the same symbol with the pirate codeword. Observe that this accusation procedure is identical to finding the "closest" among all user codewords to the pirate codeword. To put it in another way, the user codewords are random strings, but the tracing property of the CFN code guarantees that under the "marking assumption", any pirate codeword produced by a collusion of no more that t random codewords will have a small L_1-distance to one of the colluder codewords.[1] To facilitate the construction of the fuzzy locker we employ a fuzzy extractor [11] which enables one to retrieve the same random string from two different but correlated strings that have high entropy (cf. the fuzzy vault scheme [18]).

In more detail, most of the fuzzy extractors follow a correct-then-extract strategy. When the two strings are close enough, an error correcting code (ECC) can be used to eliminate their discrepancies and then a randomness extractor [7] is applied to extract a uniform output (which will later be used as a key) from the high entropy codeword (which will be the source from the point of view of the extractor). However for the fuzzy locker for CFN codes, the portion of disagreement (errors) between the codeword used for encryption and the pirate codeword extracted for decryption is quite large and beyond the decoding capability of a unique decoding ECC. We thus give up perfect correctness for the fuzzy locker, and turn to the use of list decoding [17, 30]. In a list decodable code, the error correction returns multiple candidates, but it can decode efficiently a number of errors up to portion almost 1 (as opposed to unique decoding). One last thing we need to be careful is that the rate of the ECC should be selected in a way that the entropy loss will not prohibit randomness extraction.

Combining the above tools, we will use the uniform string extracted from the fuzzy extractor as a secret key to encrypt the user secret data. We further will assume the valid messages are easily identifiable, and that decryption using a wrong key will almost never yield a valid message. These two assumptions are easy to achieve by including in the plaintext a message authentication code or a signature on the message, for details about this technique, we refer to [11, 24].

Fuzzy locker for CFN codes.: We present below the fuzzy locker for CFN codes; the choices of the parameters will be specified later. Given a randomness extractor Ext and a secure symmetric key encryption ($\mathsf{SE.Enc}, \mathsf{SE.Dec}$):

- **FL.Enc**(C, m): The algorithm inputs $C = \mathsf{c}_1 \ldots \mathsf{c}_\ell \overset{U}{\leftarrow} \mathcal{F}_q^\ell$, and message m. It first samples a random $(\ell, \kappa)_q$ Reed-Solomon code $X = x_1, \ldots, x_\ell$ which can correct up to $\ell - \ell/t$ errors, and computes $Y = \langle y_1, \ldots, y_\ell \rangle$, where $y_i = \mathsf{c}_i + x_i \mod q$; It then selects a random bitstring s and computes $k = \mathsf{Ext}(s, C)$, [2] and $c = \mathsf{SE.Enc}(k, m)$. The algorithm outputs $ct = (Y, s, c)$.
- **FL.Dec**(C^*, ct): On input a pirate codeword $C^* = \mathsf{c}_1^* \ldots \mathsf{c}_\ell^*$ and ciphertext ct, it first computes $C' = \mathsf{c}_1' \ldots \mathsf{c}_\ell'$ where $\mathsf{c}_i' = y_i - \mathsf{c}_i^* \mod q$, and it runs the list decoding algorithm on C' to get a list of RS codewords $\{X_1', \ldots, X_L'\}$. It then computes a list of possible user codewords $\{C_1, \ldots, C_L\}$ where $C_i = Y_i - X_i'$, where "-" stands for component-wise modular subtraction. The algorithm tries the following procedure for every user codeword: it computes $r_i = \mathsf{Ext}(s, C_i)$ and $m_i = \mathsf{SE.Dec}(r_i, c)$. If there exists an $m \neq \perp$, the algorithm outputs m, otherwise, it outputs $\perp$.

Security analysis. Regarding *correctness.* First we recall the basic idea of the CFN code. It randomly samples $C \leftarrow \mathcal{F}_q^\ell$. Suppose t users (w.l.o.g., we assume they have codewords $C_1, \ldots, C_t$) collude to produce a pirate codeword C^*. Due to the marking assumption, each symbol c_i^* equals to one of the corresponding symbols in $C_1, \ldots, C_t$. It follows easily that there exists a C_i, such that C^* and C_i agree on at least ℓ/t locations. We now check the decryption algorithm on $ct_i = \mathbf{FL.Enc}(C_i, m_i)$. $C' = Y - C^* = X + (C - C^*) \mod q$, thus $\{\mathsf{c}_1', \ldots, \mathsf{c}_\ell'\}$ agree with $x_1, \ldots, x_\ell$ on at least ℓ/t locations. For a Reed-Solomon code RS: $\Sigma^\kappa \to \Sigma^\ell$, it can decode at most $\ell - \sqrt{\ell\kappa}$ errors. If we have $\ell/t \geq \sqrt{\ell\kappa}$, then RS would return a list of possible candidates which contains the actual X. Then $Y - X$ would yield the user codeword C_i; correctness then follows easily.

Regarding *security*: for any honest user whose codeword C that is uniformly selected, we can think it is selected after the pirate code C^* is produced. Following the probabilistic analysis from [9], if $\ell \geq 4t\log\frac{n}{\epsilon}/3$, and $q \geq 4t$, it holds that $\Pr[C^*, C$ agree on ℓ/t locations$] \leq \epsilon$. It follows that the decoding algorithm will not return any user codeword. A bit more formally, we can think of the ciphertext (Y, s, c) as being generated following the KEM/DEM [10] framework, where Y, s encrypt the session key k which is used to encrypt the data in c. Conditioned on Y, s, the min-entropy of C can be calculated as $\ell\log q - (\ell - \kappa)\log|\Sigma|$ as s is independent of C, Y is of length ℓ, and the original random codeword has entropy $\kappa\log|\Sigma|$. Now if we have $\ell\log q - (\ell - \kappa)\log|\Sigma| \geq \Theta(\lambda)$, the strong extractor can output a sufficiently long uniform key k, thus Y, s form a secure KEM. Now the IND-CPA security of the message follows from the security of the symmetric key encryption. Due to lack of space, we defer the detailed proof in the full version.

Setting up the parameters. There are multiple constraints about selecting the parameters for the first construction of TDS from the CFN code. More specifically, for parameters $\ell, \kappa, n, t, \epsilon, \lambda$ being the dimension and degree of the RS code, the number of users, the bound of colluders, the error term in the fingerprinting code and the security parameter respectively, they have to satisfy: (1). $\ell \geq \max(\kappa t^2, 4t\log\frac{n}{\epsilon})$; (2). $\ell\log q - (\ell - k)\log\ell \geq \Theta(\lambda)$.

When $\kappa t^2 \leq 4t\log\frac{n}{\epsilon}$, we can choose $\ell = q = 4t\log n\epsilon$, and $\kappa = \Theta(\lambda)$. The resulting traitor deterring scheme will

[1]Actually, from the analysis of CFN one infers that if the pirate codeword and user codeword agree on more than ℓ/t symbols, the user can be accused.

[2]we assume here extractors can be applied to large alphabet, if not, we can simply use the bit string representing C to be the input to the extractor.

have ciphertext size $O(t^2 \log^2 \frac{n}{\epsilon})$, and the upper bound of the collusion size is $t = O(\log \frac{n}{\epsilon}/\lambda)$;

When $\kappa t^2 \geq 4t \log \frac{n}{\epsilon}$, we can choose $\ell = q = \lambda t^2$, and $\kappa = \Theta(\lambda)$. The resulting traitor deterring scheme will have ciphertext size $O(\lambda t^4)$ for any collusion size t.

To summarize, if we select the parameters in a way that all the conditions above are satisfied, then the correctness and security of the fuzzy locker for CFN code follows. Then from the general construction, we can conclude that:

Corollary 3.3 *Given PKE, there exists a TDS satisfying: fully-collusion resilient, black-box traitor deterring w.r.t. to any message distribution $\mathcal{D}$ that has min-entropy $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta - \alpha)$, where δ is the correctness probability required by the adversarial device and α is a non-negligible amount significantly smaller than δ. And it is with ciphertext size $O(\log \frac{n}{\epsilon}/\lambda)$ when $t \leq 4 \log \frac{n}{\epsilon}/\lambda$; and $O(t^4\lambda)$, if $t \geq 4 \log \frac{n}{\epsilon}/\lambda$.*

4. CONSTRUCTION FROM COMPARISON PREDICATE ENCRYPTION

In this section, we will present our second technique of constructing TDS's based on comparison predicate encryption (CPE) with an exponentially large attribute space. We first give the general construction of TDS, then instantiate the CPE from (optimized) bounded collusion functional encryption. The resulting TDS exhibits better efficiency than our CFN construction for larger traitor collusions.

4.1 TDS from CPE.

In a CPE, decryption succeeds only when $v \leq x$, where x, v are the attributes for the the ciphertext and the secret key respectively. Moreover, besides standard security, it also requires an attribute hiding property that no adversary $\mathcal{A}$ can distinguish c_0, c_1 which have attributes x_0, x_1 (assuming $x_0 < x_1$) respectively, as long as $\mathcal{A}$ does not have a secret key sk_v such that $x_0 \leq v < x_1$ (even if $\mathcal{A}$ has secret key sk_v that can decrypt both c_0, c_1). (This corresponds to the fully attribute hiding of predicate encryption [19]).

It was shown in [4,5] that a weaker version of CPE (called private linear broadcast encryption in [4], which has only a polynomially large identity space) implies a TTS. In the construction, each user is assigned an integer index as identity, and the encryption scheme has the property that $Enc(pk, i, m)$ is indistinguishable from $Enc(pk, i+1, m)$ provided $\mathcal{A}$ does not hold sk_i. Thus the tracer can do a linear scan in the identity space and feed ciphertexts generated using attributes from 0 to $n+1$ for each test. If he notices a gap between the responses for some i, and $i+1$, then the user i will be accused. The gap is guaranteed to exist as all users can decrypt $Enc(pk, n+1, m)$ and no user can decrypt $Enc(pk, 0, m)$.

To construct a TDS, we observe that if the indices are chosen randomly from an exponentially large space, they could be used as secret keys to hide the user private information. Unfortunately, it is not clear how to generalize [4,5] to an exponentially large identity space. We tackle this problem by constructing CPE's for an exponential large attribute space from functional encryption; furthermore, we apply a binary search type of tracing. In particular, in each step of search (feeding a sequence of tracing ciphertexts using a corresponding pivot identity), the recovering algorithm only consider two states for the pirate box. It is functioning, if the decryption probability is close to the claimed correctness of the pirate box; or not functioning, otherwise. Then **Rec** decides to move to a smaller pivot or a larger one.Given a CPE ($\mathsf{CPE.Setup}$,$\mathsf{CPE.KeyGen}$,$\mathsf{CPE.Enc}$,$\mathsf{CPE.Dec}$) and an authenticated encryption ($\mathsf{AE.Enc}$, $\mathsf{AE.Dec}$), our second construction of TDS (construction-II) is as follows:

- **Setup**$(\lambda, n, s_1, \ldots, s_n)$: It first runs the $\mathsf{CPE.Setup}$ algorithm to output a master key pair (mpk, msk), then it randomly selects n bitstrings $id_1, \ldots, id_n$ with length ℓ, (that is as an integer, each $id_i \in [2^\ell - 1]$), and runs the $\mathsf{CPE.KeyGen}$ algorithm to generate secret keys for the users. For user i it assigns the identity id_i and then generates the secret key sk_i =$\mathsf{CPE.KeyGen}(msk, id_i)$. It embeds the secret information of the user s_i as the ciphertext $\omega_i = \mathsf{AE.Enc}(id_i, s_i)$. The setup algorithm outputs public key mpk, secret keys $sk_1, \ldots, sk_n$, and the public parameter $para$, where $para = \langle \omega_1, \ldots, \omega_n \rangle$.
- **Enc**(mpk, m): It runs $\mathsf{CPE.Enc}(mpk, 2^\ell, m)$ and returns the corresponding output c as ciphertext.
- **Dec**(sk_i, c): This algorithm runs $\mathsf{CPE.Dec}$ with input sk_i and ciphertext c and returns m or $\perp$.
- **Rec**$^{B,\mathcal{D}}(mpk, para)$: The algorithm maintains a counter j with initial value $\ell - 1$ and repeats the following procedure until $j = 0$: It first samples a sequence of messages $m_1, \ldots, m_N$ from $\mathcal{D}$; then it generates the query ciphertexts $c_1, \ldots, c_q$, where c_i =$\mathsf{CPE.Enc}(mpk, p, m_i)$ for the position $p = 2^j$ and records how many correct answers does the box B produce; If the number of correct decryptions is more than n_0, the algorithm will set the pivot for the next test position to be $p := p - 2^{j-1}$, otherwise $p := p + 2^{j-1}$; The algorithm then decreases the counter $j := j - 1$ and repeats the procedure. The values for the parameters N, n_0 will be determined in the analysis. Suppose the above algorithm stops at position p. The **Rec** algorithm then runs $\mathsf{AE.Dec}(p, \omega_i)$ on all ω_i and returns the first non-$\perp$ value s_i.

Remark: We may implement the authenticated encryption as $\mathsf{SE.Enc}(k, s||\sigma)$ where $\sigma = Sig(s)$, where Sig is a signature scheme and the verification key is included in $para$ where $\mathsf{SE.Enc}$ is any secure symmetric key encryption scheme.

Analysis. *Correctness* and *privacy* follow straightforwardly, so we focus on the intuition of the *black-box traitor-deterring* property. Let us first present the following observations. (1). If all the colluder identities are smaller than the index p used in the tracing ciphertext, the box will decrypt correctly with probability close to δ. This holds because of the the attribute hiding property that $\mathsf{CPE.Enc}(mpk, p, m)$ is indistinguishable from the regular ciphertext $\mathsf{CPE.Enc}(mpk, 2^\ell, m)$. From this it can be deduced that failing to decrypt with probability close to δ suggests that at least one colluder identity is larger than p, thus the algorithm will not err by moving to a larger pivot. (2). Similarly, if all colluder identities are larger than the pivot index p used in the tracing ciphertext, the box will work with just negligible probability because of the payload hiding property. It follows that decrypting with a probability close to δ (non-negligible) implies at least one colluder identities is smaller than the attribute in the tracing ciphertext, and hence the tracing algorithm will not err by moving to a smaller position attribute. (see lemmas in the appendix.) The above observations imply that every move is towards a subspace containing some pirate identities.

To be a bit more formal, consider a complete binary tree which represents the whole identity space. We can think of the path walked by the **Rec** algorithm as moving along such tree. It starts from the root (represented by index $2^{\ell-1}$) and moves to the root of a complete subtree in each step. We will show via strong induction that in each move, the subtree will contain at least one colluding identity.

Theorem 4.1 *Construction-II satisfies fully collusion resilient, black-box traitor deterring property w.r.t. to any message distribution $\mathcal{D}$ with min-entropy $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta-\alpha)$ for some non-negligible α, s.t., $\delta \geq 1.5\alpha$ where δ is the correctness probability provided by the adversarial device, and the parameters $N = \alpha^{-2}\log^2\lambda$, $n_0 = (\delta - \frac{\alpha}{2})N$.*

Proof. *Correctness* follows directly from the correctness of the underlying CPE scheme. *Privacy* is also straightforward, as the user identity is uniformly sampled, the IND-CPA security of the underlying encryption scheme guarantees no information about the plaintext is leaked.

Regarding the *traitor-deterring* property: Suppose a pirate box B is created using secret keys of the users $id_1, \ldots, id_t$, and it is with $\delta-$correctness w.r.t a message distribution $\mathcal{D}$, s.t., $\mathbf{H}_\infty(\mathcal{D}) \geq -\log\delta_0$, where $\delta_0 = \delta - \alpha$, for some α. We first present three lemmas that follow easily from the payload hiding and attribute hiding properties of CPE.

Lemma 4.2 *If the underlying CFE is payload hiding, and the tracing ciphertext C is created using a pivot p and message m randomly sampled from $\mathcal{D}$, and $id_i > p$ for all $i = 1, \ldots, t$, then: $|\Pr[B(C) = m] - \delta_0| = \mathsf{negl}(\lambda)$.*

Lemma 4.3 *If the underlying CFE is attribute hiding, and two tracing ciphertexts C_1, C_2 are created using message m, and pivots p_1, p_2 respectively, and for all $i = 1, \ldots, t$, $id_i \notin [p_1, p_2)$, then: $|\Pr[B(C_1) = m] - \Pr[B(C_2) = m]| = \mathsf{negl}(\lambda)$.*

Lemma 4.4 *If the underlying CFE is attribute hiding, the tracing ciphertext C is created using a message m randomly sampled from $\mathcal{D}$ and a pivot p, and $id_i \leq p$ for $i = 1, \ldots, t$, then $|\Pr[B(C) = m] - \delta| = \mathsf{negl}(\lambda)$.*

We then estimate the parameters n_0, N for determining whether B works. Following the Chernoff bounds, $\Pr[X < (1-\omega)\mu] \leq e^{-\omega^2\mu/2}$, and $\Pr[X > (1+\omega)\mu] \leq e^{-\omega^2\mu/3}$, when $X = \sum X_i$, $\{X_i\}$ are independent random variables over $\{0,1\}$, $0 < \omega < 1$, and $\mu = E(X)$. In this setting, X_i is the event denoting when **Rec** feeds the $i-$th ciphertext which encrypts a random message m sampled from $\mathcal{D}$, the box B returns the plaintext correctly. It follows that, if the traitor indices are all smaller than the pivot, B works with δ-correctness, $\Pr[X_i = 1] \geq \delta$. After repeating N times, the probability that at most $n_0 = (\delta - \frac{\alpha}{2})N$ correct answers are returned by B is bounded by $e^{-\alpha^2 N/8}$. On the other hand, if the traitor indices are all larger than the pivot, B works with only probability $\delta - \alpha$. The probability that B returns more than n_0 correct answers is bounded by $e^{-\alpha^2 N/12}$.

Setting parameters $N = \alpha^{-2}\log^2\lambda, n_0 = (\delta - \frac{\alpha}{2})N$, less than n_0 correct answers means that there must be a traitor index larger than the pivot; more than n_0 correct answers means there must be a traitor index smaller than the pivot.

Now we are ready to proceed to prove the theorem. We can represent all users as leaves in a complete binary tree indexed by $\{1, \ldots, 2^\ell\}$; given this **Rec** moves a pivot performing a binary search in this tree by selecting a sequence of subtrees $S_0, S_1, \ldots$ in the following fashion: at move $j \geq 1$, the pivot p_j defines the subtree S_{j-1} as the subtree of the complete binary tree that is rooted at a node v that has p_j as the index of the rightmost leaf of the left subtree of S_{j-1}. Observe that S_0 is the whole tree. We will prove by strong induction that for all $j \geq 0$, S_j contains a traitor. The base, $j = 0$, is straightforward. Suppose that the statement is true for $S_0, S_1, \ldots, S_{j-1}$. We will prove for S_j.

Case 1. Suppose that S_j is a left subtree of S_{j-1}. This means that there is a traitor with index at most p_j (otherwise, if all traitors had a bigger index, then by lemma 4.2 the pirate box would be unsuccessful and the recovering algorithm would move to the right subtree of S_{j-1}). Now suppose that none of the traitors belong to S_j and let u be the largest index of a traitor that has index at most p_j. By the fact that u does not belong to S_j we know that at least one of the subtrees $S_1, \ldots, S_{j-1}$ is a right subtree of its containing parent subtree. Let S_k be such a subtree with the largest $k \leq j-1$. Now note that when the recovering algorithm used pivot p_k (which lies in the center of subtree S_{k-1}) it holds that: $u \leq p_k$. Observe that there is no traitor with index in the set $\{p_k + 1, \ldots, p_j\}$. Based on lemma 4.3 the decision of **Rec** when testing with pivot p_j and pivot p_k should be the same (with overwhelming probability). This leads to a contradiction as **Rec** moved to the right (resp. left) when testing with index p_k (resp. p_j).

Similarly, we can argue for the case that S_j is a right subtree of S_{j-1}. We can conclude that S_ℓ is a single leaf node and it also denotes a traitor. $\square$

4.2 Instantiations of CPE.

Next we will give concrete constructions of CPE supporting an exponentially large attribute space. We first note that, a straightforward instantiation can be obtained from general functional encryption (FE) which can be constructed using indistinguishability obfuscation (iO) [13]. The resulting TDS will have only a constant size ciphertext however it will rely on assumptions related to multilinear maps [13].

We now present an instantiation from standard assumptions. Note that there exists a bounded collusion FE from standard assumptions. In a TDS there is only a potentially small (and in any case polynomially bounded) subset of users that is colluding to produce a pirate box. We show how to construct a CPE from bounded collusion FE.

Instantiation-I. General FE secure for a single key query with succinct ciphertext was constructed in [14]. To amplify [14] to a q-query secure FE, one simply runs q independent 1-query secure FE schemes in parallel. Each secret key is generated using a different master secret key (this step will require that the authority maintains and updates a private state to keep track of which master secret keys have been used), while each master public key will be used to encrypt the message resulting in a vector of q ciphertexts encrypting the same message. Unfortunately using this scheme to instantiate the CPE for a TDS would force $q = n$. To see this, even if we choose $q = n-1$, there exist a pair of users i, j such that their secret keys are generated using a same master secret key (say the k-th master secret key). When user i, j are corrupted together, no security can be guaranteed for the k-th 1-query secure FE instance, and the CPE scheme cannot be shown secure. Thus the resulting TDS will have ciphertext size $O(n \cdot \text{poly}(\lambda))$ which is not preferable especially given that the collusion t might be much smaller than n. We then show how to improve the ciphertext complexity.

Instantiation-II. A stateless q bounded FE was constructed in [15] from a 1-query secure FE using techniques from secure computation, and their scheme guarantees security under arbitrary collusion with size q, even if more keys are issued (say n). We can use such a t-bounded FE to instantiate a CPE facing t corrupted users. Unfortunately, the parameters in [15] were chosen in a way that the ciphertext size is as big as $O(D^2t^6\lambda\tau)$, where D is the maximum degree of the polynomial representing the circuits describing the functions that FE supports, and τ is the ciphertext size of the underlying 1-query secure FE. For some parameters d, N, in the construction, there are N 1-query secure FE instances. The encryption algorithm will do a $(d+1, N)$ secret sharing on the message and will encrypt each share independently under the N 1-query FE instances. Each user will be assigned a random subset (denoted by Γ_i, and $|\Gamma_i| = dD+1$) of keys each of which is generated using the corresponding master secret key. Note that prior to encrypting each share is padded with additional randomness to ensure the simulation can succeed. In total there are N ciphertexts each encrypting $O(t^2\lambda)$ plaintext elements. (See [15] for details.)

Reference [15] requires that the collusion of size t can not break enough 1-query secure FE instances to get $d+1$ shares and obtain extra information about the message. More specifically, it requires $|\cup_{i\neq j}(\Gamma_i \cap \Gamma_j)| \leq d$. We observe that if we can replace this condition to be $|\cup_{i_1,\ldots,i_a}(\cap_{i_j=i_1}^{i_a}\Gamma_{i_j})| \leq d$, for any integer $a \geq 2$, through a probabilistic analysis, we can bring down N to $O((Dt)^{1+e}\lambda)$ (for $e = 1/(a-1)$). Doing this optimization requires us to use an a-query secure FE as the underlying building block. We can obtain a succinct a-query secure FE for some polynomially bounded a by applying the technique of [15] to the succinct 1-query secure FE of [14]. In this way we obtain a a-query FE that has ciphertext size $O(\text{poly}(\lambda))$ which is independent from the number of users. Then we can apply the extended probabilistic analysis explained above and to obtain a t-query FE with ciphertext $O(t^{3+e}\text{poly}(\lambda))$. Note that we are using circuits for the comparison predicate only and thus the degree D of the polynomial representing the circuits is at most λ.

Our CPE instantiation. Our final CPE instantiation will be a hybrid of the two instantiations above. When $t \leq n^{\frac{1}{3+e}}$, we use *instantiation-II*, the optimized t-FE; when $t > n^{\frac{1}{3+e}}$, we simply use *instantiation-I* of n-query secure FE. The resulting TDS will be with ciphertext size $\min[O(t^{3+e} \cdot \text{poly}(\lambda)), O(n \cdot \text{poly}(\lambda))]$. As the succinct 1-query FE can be built on fully homomorphic encryption [6] and attribute based encryption [16], both of which can be based on the LWE assumption [29] efficiently. We summarize the above, and refer detailed analysis to the appendix.

Corollary 4.5 *Under the subexponential LWE assumption, there exists a TDS satisfying: fully collusion resilient, black-box traitor deterring w.r.t to any message distribution $\mathcal{D}$ with $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta-\alpha)$ for some non-negligible α, where δ is the correctness probability provided by the pirate box and $\delta \geq 1.5\alpha$, and the parameters $N = \alpha^{-2}\log^2\lambda$, $n_0 = (\delta - \frac{\alpha}{2})N$. It has ciphertext length $\min[O(t^{3+e} \cdot poly(\lambda)), O(n \cdot poly(\lambda))]$, where n, t are total number of users and corrupted users, $e = 1/poly(\lambda)$, and λ the security parameter.*

5. TRAITOR DETERRING IN THE KNOWN CIPHERTEXT MODEL

In the known ciphertext model for TDS, the adversary has a weaker goal: it aims to produce a pirate box that works w.r.t. a *given* sequence of ciphertexts.

Because the sequence of ciphertexts is fixed there is a trivial way to implement the pirate decoder: simply store a database of all the plaintexts. Thus, in the known ciphertext model, the adversary should only win when the size of the decoder is smaller than the trivial attack; formally, we will associate an attack in this model with a "space rate" that is equal to the size of the pirate box divided by the length of the total plaintext contained in the known ciphertext. An ideally secure scheme should work with respect to any space rate $o(1)$.

The known ciphertext model is applicable to the setting of distributing content via CDs, or granting access to an encrypted database, since in these cases, the attack occurs after the target ciphertexts become known. (In contrast, the original traitor deterring model is applicable to all other settings, e.g., online streaming, and movie distribution in Pay-TV etc.). Traitor deterring in the known ciphertext model reformulates in the black-box public-key setting the problem of constructing digital signets as posed in [12]. In [12] a construction for any space rate $o(1)$ is presented however it requires the unfalsifiable assumption that the function $f(x) = g_1^x||g_2^x||\ldots||g_\ell^x$ is incompressible (as well as they assume non-black-box recoverability). They leave as open question whether a construction exists that is secure under a falsifiable assumption; using our TDS's we resolve this open question in this section.

5.1 Definition: the known ciphertext model.

We provide the formal definition of the known ciphertext model that strengthens our traitor deterring definition.

$(1-\epsilon)$-correctness. Since the pirate box B may work only for the fixed set of ciphertext $SC = \{c_1, \ldots, c_n\}$, we require for SC, it almost always works, i.e., $\Pr[B(i, c_i) = m_i] \geq 1 - \mathsf{negl}(\lambda)$, where m_i is the $\Theta(\lambda)$ bit plaintext that is encrypted in c_i (note we also allow B to receive the index of c_i).

Privacy: This is the same as in section 2.

Traitor Deterring for Known Ciphertexts. The main difference with the traitor deterring property is that the adversary is aware of the ciphertexts before making the device B, and hence can embed some information into B so that B is able to check the decryption queries and only works for the given ciphertexts. Formally,

- The challenger $\mathcal{C}$ simulates the **Setup** algorithm and the adversary $\mathcal{A}$ receives pk. $\mathcal{A}$ then sends to $\mathcal{C}$ a vector of secret information $s_1, \ldots, s_n$, an arbitrary subset $T \subseteq \{1, \ldots, n\}$ as well as a distribution $\mathcal{P}_k$ with support set that contains k-long vectors of plaintexts for some $k = O(\text{poly}(\lambda))$. [3] $\mathcal{A}$ receives the secret keys of all users in T, $\{sk_i \mid i \in T\}$ as well as the public parameter $para$.
- $\mathcal{C}$ samples $(m_1, \ldots, m_k)$ from $\mathcal{P}_k$ and sends $\mathcal{A}$, the sequence of ciphertexts $SC = \langle c_1, \ldots, c_k\rangle$ where $c_i = \mathbf{Enc}(pk, m_i)$; finally, $\mathcal{A}$ outputs an implementation B.
- $\mathcal{C}$ outputs 1 if and only if $\mathbf{Rec}^B(pk, para) \notin \{s_{i_1}, \ldots, s_{i_t}\}$.

We denote the event that $\mathcal{C}$ outputs 1 in the above game by $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{KCdeter}}(1^\lambda)$. We say a scheme achieves black-box traitor deterring for known ciphertexts with space rate $\mathsf{s}(k, \lambda)$ if for any PPT adversary $\mathcal{A}$,

[3]This includes the case of encrypting one single long message (e.g., a movie file): it is first divided into k blocks and each block is encrypted individually.

$$\Pr[\text{B is } (1-\epsilon)\text{-correct w.r.t } SC \wedge \frac{|B|}{k\lambda} \leq \mathsf{s}(k,\lambda) \wedge \mathsf{Succ}_{\mathcal{A}}^{\mathtt{KCdeter}}(1^\lambda)]$$

is a negligible function on λ, where $|B|$ denotes the size of the program B. Note that for $\mathsf{s}(k,\lambda) = \Theta(1)$ it is trivial to construct a device B that allows the adversary to win the above game — simply store all plaintexts $m_1, \ldots, m_k$ in B. Thus, the question that is raised is whether it is possible to deter with space rate that is $o(1)$.

5.2 Feasibility and infeasibility for the known ciphertext model.

At first sight, it may seem impossible to have a black-box recovering algorithm in the known ciphertext setting, since the **Rec** algorithm is restricted by the fact that the adversarial box is only guaranteed to work for a fixed set of ciphertexts. Indeed, although the size of B can be smaller than the size of the ciphertexts it is supposed to work for, there are ways for the adversary to embed some information and check whether a submitted ciphertext belongs to the targeted sequence, while reject all other ciphertexts submitted to it. We formalize this intuition and we show a simple attack following this principle that rules out the possibility of black-box traitor deterring for known ciphertexts for a range of space rates. However, we also observe that in order for the box B to perform a membership test in the targeted ciphertext sequence, the false positive probability of the testing algorithm increases as the storage used by B gets smaller. When the false positive probability becomes sufficiently high, a random sample of ciphertext will be answered by the box B with non-negligible probability δ, and thus B becomes a $\delta-$correct box in the regular model (as defined section 2); in this way, we can still apply our constructions of TDS's against known ciphertext type of attacks. For ease of presentation, we consider only the 1-correct case, while all results will also follow for the case of $(1-\epsilon)$-correctness.

The intuition behind the proof of the following theorem is that when the suitable space bound is imposed on the pirate device, it will have to resort to using the secret-key in a sufficiently large plaintext distribution that can be sampled with a non-negligible probability from the plaintext space. As a result, the decryption box, is a general purpose decryption box that is δ-correct for some non-negligible δ and thus our recoverability algorithms developed for traitor deterring can be applied in the known ciphertext model as well.

Theorem 5.1 *There exists a TDS with superpolynomial in λ plaintext space that satisfies black-box traitor deterring for known ciphertexts with space rate $\mathsf{s}(k,\lambda) = O(\log(\lambda)/\lambda) = o(1)$ for any $k = \Omega(\lambda)$.*

PROOF. We will show that a TDS satisfying black-box traitor deterring with any pirate box with λ^{-c}-correctness is also a TDS with black-box traitor deterring in the known ciphertext model for any $c \in \mathbb{N}$ for the stated space rate.

First, we recall a lower bound of the false positive probability in the approximate membership testing problem (see definition A.1 in the appendix) fwhen the space of the tester is upper bounded. For a universe U with size u, and $V \subset U$ with size v, and $v \ll u$, using space τ, the false positive η of any membership tester satisfies $2^\tau \leq (2\eta)^v$. (see Lemma A.2 in the appendix).Applying logarithm to both sides, we can get $\eta \geq 2^{-\frac{\tau}{v}-1}$, thus if $\tau \leq c \cdot v \cdot \log\lambda$, we have $\eta \geq \lambda^{-c}$.

Next, we will use the above result to show that a useful decryption box B with size $O(k \cdot \log\lambda)$ will have non-negligible correctness w.r.t. uniform distribution over the message space. Specifically, we will build an approximate membership tester T (using B) for $V = \{(m_1, c_1), \ldots, (m_k, c_k)\}$, a subset of the universe U of all plaintext/ciphertext pairs, with a similar storage as follows. Whenever queried a uniformly random pair (m, c), T queries B with c, if B outputs m, T outputs 1, otherwise T outputs 0. It is easy to see that if $(m, c) \in V$, T always accepts; if $(m, c) \notin V$, T accepts with probability δ, where $\delta = \Pr[B(c) = m \wedge (m, c) \notin V]$. Furthermore, T only needs an extra storage of $O(\lambda)$ bits to store the query and compare whether the answer of B is valid. In the setting that $k = \Omega(\lambda)$, the storage of T is still $O(k \cdot \log(\lambda))$. Observe that if δ is negligible, T is a membership tester which violates the bound in Lemma A.2.

With the above claim, we can see that for a randomly sampled ciphertext, the box B will answer with some probability δ and thus we can run the **Rec** algorithm and retrieve the corresponding secret information of one of the colluders assuming that the TDS works for δ w.r.t any distribution $\mathcal{D}$ for which it holds that $\delta \geq 2^{-H_\infty(\mathcal{D})} + \alpha$ where α is an arbitrary non-negligible function. $\square$

Impossibility results. Next we will show that the above bound of the size of B is essentially tight, by describing a generic attack against any traitor deterring scheme for known ciphertexts. The attacking strategy is simple: using Bloom filters [1] the adversary produces a box that contains a membership tester built in so that it will answer only when the decryption query belongs to the ciphertext set. This makes two boxes implemented using different keys indistinguishable via only oracle access, thus black-box recoverability will contradict privacy in this setting. For details of the proof we refer to the appendix.

Proposition 5.2 *There does not exist any, even 1-resilient, black-box TDS in the known ciphertext model for space rate $\mathsf{s}(k,\lambda) = \Omega(\log^2\lambda/\lambda)$ for any k.*

6. USING BITCOIN AS COLLATERAL

Bitcoin is a decentralized cryptocurrency [25] that uses a *publicly* maintained ledger to store transactions and record transfers between bitcoin accounts. Each bitcoin account is essentially a hash of a public-key and the owner of the secret-key has the ability to transfer funds from the account by posting a transaction in the bitcoin network that contains a signature generated by the account's secret-key. The characteristic of bitcoin accounts is that the secret-keys represent complete ownership of the account.

We consider a TDS deployment for a broadcast service where a service provider (SP) wants to use a certain amount of bitcoin as collateral. Upon initiation of the service the SP generates bitcoin accounts corresponding to each of the n users setting $s_i = (a_i, k_i)$ where a_i is the bitcoin address and k_i is the associated secret-key. When a user joins the system it requests from the user to transfer some amount of x bitcoin to the a_i bitcoin account. The SP shares the account information (a_i, k_i) with the user so that it is ensured that the x bitcoin is a collateral and the user has the option to obtain the collateral back whenever she wishes (and cancel her subscription). The SP then embeds s_i into the public directory. At the same time the SP gives to the user the secret-key sk_i that corresponds to the account, and sets a service agreement that the account should be "frozen" such that no outgoing transaction is allowed until the user unsubscribes the service. The user from this point on can use the service and decrypt ciphertexts associated with the service. In regular intervals the SP checks the public ledger to see whether any active account has an outgoing transaction (no matter to who is transferred). If there is such a case the subscription of the user should be cancelled (this would require the revocation of the key sk_i an issue that we do not explicitly deal here but can be handled generically via e.g., a re-key operation where the SP at regular intervals refreshes the keys of the system keeping the same collaterals for all the remaining subscribers). Observe that due to the properties of TDS for as long as the user respects the service agreement and does not share her secret-key her collateral bitcoin remain safe. The user can collect her collateral bitcoin whenever she wants to terminate the service.

7. CONCLUSION AND OPEN PROBLEMS

We formalize and construct the new cryptographic primitive of TDS that achieves proactive deterrence of unauthorized device distribution and we show how bitcoin can be used as a collateral for a TDS deployment.We also revisit the open problem of digital signets and reformulate as TDS in the known ciphertext model, and show how we can utilize TDS to solve it under parameter choices that allow a possibility result.

There are many interesting open problems that remain. The first one is how to construct a TDS with constant size ciphertext under standard assumptions. This may require a fuzzy locker for, e.g., Tardos code [31] which currently uses a secret tracing algorithm. Also, a construction of unbounded collusion secure CPE is another alternative which will be of independent interest. Furthermore, combining a TDS with a revocation system as in [27] to obtain a “Trace Deterring and Revoke scheme” would be an important advance.

Acknowledgements This work was supported by European Research Council Project CODAMODA.

8. REFERENCES

[1] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[2] D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In *Advances in Cryptology - CRYPTO ’99*, pages 338–353, 1999.

[3] D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In *ACM CCS 2008*, pages 501–510.

[4] D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT 2006*, pages 573–592.

[5] D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM CCS 2006*, pages 211–220, 2006.

[6] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS 2011*, pages 97–106.

[7] L. Carter and M. N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[8] H. Chabanne, D. H. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT 2005*, pages 542–558, 2005.

[9] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *CRYPTO 94*, pages 257–270, 1994.

[10] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2004.

[11] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*, pages 523–540, 2004.

[12] C. Dwork, J. B. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information (preliminary version). In *STOC*, pages 489–498, 1996.

[13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS 2013*, pages 40–49, 2013.

[14] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC’13*, pages 555–564.

[15] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO 2012*, pages 162–179, 2012.

[16] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from lwe. *IACR Cryptology ePrint Archive*, 2015.

[17] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Trans on Information Theory*, 45(6):1757–1767, 1999.

[18] A. Juels and M. Sudan. A fuzzy vault scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006.

[19] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. EUROCRYPT’08, pages 146–162.

[20] A. Kiayias and S. Pehlivanoglu. *Encryption for Digital Content*, volume 52 of *Advances in Information Security*. Springer, 2010.

[21] A. Kiayias and Q. Tang. How to keep a secret: leakage deterring public-key cryptosystems. In *ACM CCS 2013*, pages 943–954.

[22] A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In *EUROCRYPT’02*, pages 450–465.

[23] K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In *Advances in Cryptology - EUROCRYPT ’98*, pages 145–157, 1998.

[24] S. Micali, C. Peikert, M. Sudan, and D. A. Wilson. Optimal error correction against computationally bounded noise. In *TCC 2005*, pages 1–16, 2005.

[25] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009.

[26] M. Naor. On cryptographic assumptions and challenges. In *CRYPTO 2003*, pages 96–109, 2003.

[27] M. Naor and B. Pinkas. Efficient trace and revoke schemes. FC ’00, pages 1–20.

[28] A. Pagh, R. Pagh, and S. S. Rao. An optimal bloom filter replacement. In *SODA 2005*, pages 823–829.

[29] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[30] M. Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.

[31] G. Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2), 2008.

APPENDIX

A. SOME OMITTED PROOFS

Proof of corollary 4.5. We first analyze the parameters of the instantiation of CPE in section 4.2.

For the parallel repetition construction, as every secret key is generated from a different master secret key, and the ciphertext $CT = \{CT_i\}$, and $CT_i = \mathsf{OneFE.Enc}(mpk_i, m)$ for the same message m. It is obvious that due to the security of the underlying 1-query secure FE, every ciphertext is simulatable given $C(m)$ by running the corresponding 1-query secure FE simulator.

Next, we will analyze the optimized q-query secure FE. Following the analysis of [15], the only difference of our scheme is that we use a c-query secure FE as the building block. We replace the first restriction to be $| \cup_{i_1,\ldots,i_c} (\cap_{i_j=i_1}^{i_c} \Gamma_{i_j})| \leq d$, as each instance now can assure security given that the for less than d instances, the collusion of t users have c keys.

Now we can analyze that this condition will improve the size of N. To be more specific, suppose X_{ij} denotes the expected size of the intersection of two random subsets X_i, X_j,

$$E(X_{ij}) = \sum_X E(X_{ij}|X_i = X)\Pr[X_i = X] = E(X_{ij}|X_i = X).$$

To see this, for any X with size $dD+1$, the expected value $E(X_{ij}|X_i = X)$ is the same and the conditional distribution follows the hypergeometric distribution with $dD+1$ good balls and $tD+1$ draws, thus $E(X_{ij}|X_i = X) = (tD+1)^2/N$.

For X_{ijk} denoting the expected size of intersection of three random subsets X_i, X_j, X_k, we have:

$$\begin{aligned} E(X_{ijk}) &= \sum_i \Pr[X_{ij} = i] E(X_{ijk}|X_{ij} = i) \\ &= \sum_i \Pr[X_{ij} = i] i \cdot (dD+1)/N \\ &= \frac{(dD+1)}{N} E(X_{ij}) = \frac{(tD+1)^3}{N^2}. \end{aligned}$$

Similarly, we can generalize the second formula to the expected size of intersection of c random subsets which is $(tD+1)^c/N^{c-1}$.

The expected size E_c of the disjoint of all possible intersection of c subsets: $| \cup_{i_1,\ldots,i_c} (\cap_{i_j=i_1}^{i_c} \Gamma_{i_j})|$ (we denote as γ) is the summation of all combinations of $X_{i_1,\ldots,i_c}$, i.e.,

$$E_c = q(q-1)\ldots(q-c+1)\cdot(dD+1)^c/N^{c+1} \leq (2qdD)^c/N^{c-1}.$$

If we let $E_c \leq \frac{d}{2}$, i.e. let $N = 4d(Dq)^{1+e}$, where $e = 1+1/c$, then following the Chernoff bound: for any $\delta > 0$, $\Pr[X > (1+\delta)E[X]] \leq e^{\frac{-\delta^2}{2+\delta}E[X]}$, thus:

$$\Pr[\gamma \geq t] = \Pr[\gamma \geq (1+1)E_c] \leq e^{-\frac{E_c}{3}} = e^{-d/6}.$$

While we are focusing on comparison predicate, which can be easily implemented e.g., for two numbers x, v represented using bits $x_1 \ldots x_\ell, v_1 \ldots v_\ell$, the comparison predicate $[x \leq v] \iff [x = v] \vee [x_1 = 0 \wedge v_1 = 1] \vee [(x_1 = v_1) \wedge (x_2 = 0 \wedge v_2 = 1)] \vee \ldots \vee [(x_1 \ldots x_{\ell-1} = v_1 \ldots v_{\ell-1}) \wedge x = v]$, is with degree at most $\ell = O(\lambda)$.

Summarizing the above analysis, if we set $d = \lambda$, $N = O(q^{1+e}\text{poly}(\lambda))$, we have $\Pr[| \cup_{i_1,\ldots,i_c} (\cap_{i_j=i_1}^{i_c} \Gamma_{i_j})| \geq d] \leq e^{-\lambda/6} = \mathsf{negl}(\lambda)$. Then following the analysis of [15], all the ciphertexts can be simulated.

The ciphertext efficiency of the optimized scheme is that $O(N \cdot S \cdot \tau) = O(q^{3+e} \cdot \text{poly}(\lambda))$, where N, S are the number of ciphertext and plaintext elements in each ciphertext respectively, and τ is the size of ciphertext for each plaintext element of the underlying succinct $c-$bound FE, for $c = \text{poly}(\lambda)$, arbitrary polynomially bounded integer.

On the other hand, the parallel repetition based construction is simply with ciphertext efficiency $O(n \cdot \text{poly}(\lambda))$.

As the succinct 1-query secure FE can be constructed assuming succinct fully homomorphic encryption and attribute based encryption for circuit, and the following two can be based on LWE assumption. Then following theorem 4.1, we can conclude as in the corollary.

Definition A.1 [1,7] *For a subset V randomly chosen from a universe U, the approximate membership testing problem with a false positive probability η is to produce a data structure T such that, for a random element $x \in U$, if $x \in V$, $T(x)$ always outputs 1, while if $x \notin V$, $T(x)$ outputs 0 with probability at least $1-\eta$ (i.e., it may output 1 with probability at most η, a false positive).*

Lemma A.2 *[28]. For a universe U with size u, and $V \subset U$ with size v, and $v \ll u$, using space τ, the false positive η in the AMT problem satisfies $2^\tau \leq (2\eta)^v$.*

Proof of proposition 5.2. We will show an attack that uses up to $\Omega(k \cdot \log^2 \lambda)$ space can defeat the black-box traitor deterring if the privacy of the user data is also required.

The adversary selects a set k of distinct plaintexts $S = \{m_1, \ldots, m_k\}$ and submits this as a distribution $\mathcal{P}_k$. The adversary furthermore corrupts user i and receives the corresponding set of ciphertexts $S = \{c_1, \ldots, c_k\}$. $\mathcal{A}$ creates a membership tester T with a false positive probability ϵ for S, such that T can be constructed using space $O(k \log \frac{1}{\epsilon})$. Using Bloom filters, [1,28], if $\mathcal{A}$ uses space $\Theta(k \log^2 \lambda)$, then ϵ will be a negligible function in λ. $\mathcal{A}$ produces a pirate box B with T built in, and when is given input c, B first checks whether $c \in S$ (besides the storage, this checking program has only a constant description). Assuming the test passes, the algorithm applies the key sk_i to decrypt the ciphertext.

Assume there is a black-box recovering algorithm recovers user i's secret information when given oracle access to B, and another adversary corrupts a different user j and build a similar box B' s.t., it is the same as B only when inputs passes the tester, B' uses sk_j to decrypt the query and respond. It is obvious that the input/output distribution of B, B' is statistically close (with the only difference because of the negligibly small false positive), these two boxes cannot be distinguished by any algorithm via only oracle access to them. Thus the **Rec** algorithm will also return the same output, i.e., secret information of user i when having oracle access to B', hence contradicting the privacy property.