Traitor-Tracing from LWE Made Simple and Attribute-Based

Yilei Chen[1(✉)], Vinod Vaikuntanathan[2], Brent Waters[3], Hoeteck Wee[4], and Daniel Wichs[5]

[1] Visa Research, Palo Alto, USA
yilchen@visa.com
[2] MIT, Cambridge, USA
vinodv@csail.mit.edu
[3] The University of Texas at Austin, Austin, USA
bwaters@cs.utexas.edu
[4] CNRS, ENS, PSL, Paris, France
wee@di.ens.fr
[5] Northeastern University, Boston, USA
wichs@ccs.neu.edu

Abstract. A traitor tracing scheme is a public key encryption scheme for which there are many secret decryption keys. Any of these keys can decrypt a ciphertext; moreover, even if a coalition of users collude, put together their decryption keys and attempt to create a new decryption key, there is an efficient algorithm to trace the new key to at least one the colluders.

Recently, Goyal, Koppula and Waters (GKW, STOC 18) provided the first traitor tracing scheme from LWE with ciphertext and secret key sizes that grow polynomially in $\log n$, where n is the number of users. The main technical building block in their construction is a strengthening of (bounded collusion secure) secret-key functional encryption which they refer to as mixed functional encryption (FE).

In this work, we improve upon and extend the GKW traitor tracing scheme:

- We provide simpler constructions of mixed FE schemes based on the LWE assumption. Our constructions improve upon the GKW construction in terms of expressiveness, modularity, and security.
- We provide a construction of attribute-based traitor tracing for all circuits based on the LWE assumption.

1 Introduction

A traitor tracing scheme [14] is a public key encryption scheme for which there are many secret decryption keys, so that any of these keys could decrypt the ciphertext. In addition, if a coalition of users collude to create a new decryption key, then there is an efficient algorithm to trace the new key to (at least one of) its creators.

A. Beimel and S. Dziembowski (Eds.): TCC 2018, LNCS 11240, pp. 341–369, 2018.
https://doi.org/10.1007/978-3-030-03810-6_13

Recently, Goyal, Koppula and Waters (GKW) [20] constructed the first traitor tracing scheme from standard assumptions with ciphertext and secret key sizes that grow polynomially in $\log n$, where n is the number of users. The security of the scheme relies on the polynomial hardness of the LWE assumption with sub-exponential modulus-to-noise ratio. The main technical building block in their construction is a strengthening of bounded-collusion-secure secret-key functional encryption which they refer to as *mixed functional encryption* (mixed FE), and the bulk of the paper (over 60 pages) is dedicated to constructing mixed FE for branching programs.

Mixed FE. A functional encryption (FE) scheme allows us to encrypt a program f and create secret keys for inputs x, so that given an encryption of f and a key for x, we learn $f(x)$ and nothing else about f. In this work, we focus on secret-key FE schemes where encryption requires the master secret key, and security is guaranteed for an *a-priori bounded number of ciphertexts*, but an unbounded number of secret keys. A mixed FE scheme is a secret-key FE scheme with an additional "restricted" public-key encryption algorithm that enables encrypting only the "all accept" program; roughly speaking, we can obliviously sample encryptions of the "all accept" programs without knowing the master secret key.

This Work. In this work, we improve upon and extend the GKW traitor tracing scheme:

- We provide simpler and more modular constructions of mixed FE schemes based on the LWE assumption. Our constructions improve upon the GKW construction in terms of both expressiveness and security. Our first construction obtains mixed FE for all circuits and with adaptive security, whereas the prior construction [20] only achieves selective security for branching programs. Our second construction achieves selective security for all circuits with tighter overhead growth for the number of secret key ciphertexts generated.
- We provide a construction of attribute-based traitor tracing schemes for all circuits based on the LWE assumption.

1.1 Technical Overview

In the technical overview, we focus on our simpler constructions of mixed FE schemes. See Fig. 1 for a brief summary. In addition to the algorithms (Setup, SKGen, SK-Enc, Dec) in a standard secret-key FE scheme, a mixed FE scheme has an additional PK-Enc algorithm that is able to encrypt "all-1" program without knowing the master secret key.

Both of our new constructions work for any arbitrary polynomial bound t on the number of ciphertexts. The GKW construction focused on the setting

Construction	Function class	blow-up	security
[20]	NC^1	$O(t^2)$	selective
Lockable obfuscation + [18]	poly-size circuits	$O(t^4)$	adaptive
Lockable obfuscation + [4]	poly-size circuits	$O(t^2)$	selective
Key-homomorphic PCPRF	poly-size circuits	$O(t)$	selective

Fig. 1. Summary of our t-CT mixed FE schemes. PCPRF refers to private constrained PRF. Here, selective means that all t ciphertexts queries come before the (unbounded) secret key queries, whereas adaptive allows arbitrary interweaving of these queries. Note that "FULL-SIM security" in [4] correspond to selective security here (in Definition 2.2 for FULL-SIM in the paper, secret keys in the security game correspond to ciphertexts in our setting). As noted earlier, the work of [20] proved security for the case of $t = 2$ and the more general case with $O(t^2)$ blowup was only sketched.

$t = 2$ which already suffices for traitor-tracing, and provided a brief sketch for extending the construction to arbitrary t but without any analysis.[1]

We provide two constructions achieving incomparable guarantees, based on two natural and complementary approaches:

1. The first construction shows how to generically transform a t-CT secret-key functional encryption (SKFE) into a t-CT mixed FE using lockable obfuscation (a.k.a. "compute-and-compare obfuscator") [19,28], which can be based on LWE. This construction extends the coverage of mixed FE in [20] from branching programs to all circuits. It also carries over the adaptivity achieved by the underlying t-CT SKFE schemes (e.g. in [2,4,18][2]) to the final mixed FE scheme. A t-CT SKFE schemes can be constructed from any one-way function [18,27]; thus, this construction shows how to leverage lockable obfuscation to add a "restricted public-key mode" to any t-CT SKFE scheme and give us a mixed-FE scheme. The construction and proof fit in a little over a page.
2. The second construction starts from the observation that the LWE-based private-constrained PRFs in [11–13] already give a 1-CT mixed FE scheme. Furthermore, we show how to construct a t-CT mixed FE in a natural way leveraging the key-homomorphic property of the private constrained PRFs. Therefore we get a construction of t-CT mixed FE for circuits for which security follows directly from the key-homomorphic PCPRF.

[1] In this work we use a simulation based definition of security where t refers to the (maximum) total number of ciphertexts seen by the attacker. The work of [20] uses an indistinguishability notion of security where they refer to the number of encryption oracle queries given to the attacker in addition to the challenge ciphertext. Roughly, a t ciphertext scheme in our definition corresponds to a $t - 1$ query scheme in [20].

[2] These prior works construct 1-CT t-SK public-key FE scheme, which implies a many-CT t-SK public-key FE scheme, and therefore a many-CT, t-SK secret-key FE scheme. By flipping the ciphertexts and secret keys, we obtain a t-CT, many-SK SKFE.

The blow-up of our t-CT mixed-FE is only $O(t)$. Previously for simulation-secure secret-key FE with bounded collusion, the blow-up is at least $O(t^2)$ [2,4] (let us remark that these constructions are public-key FE schemes). We sketch this construction and proof later in the introduction, again in a little over a page.

1.2 Mixed FE from Lockable Obfuscation

Our first construction adds lockable obfuscation on top of a plain t-CT SKFE to produce the public-key ciphertext, i.e. let the public-key ciphertext be the dummy obfuscated programs that always evaluate to "$\perp$".

In more detail, we construct the mixed-FE scheme as follows:

- Setup: Choose a master secret key (msk) for the SKFE.
- SKGen(msk, x): use the SKFE msk to generate sk_x.
- SK-Enc(msk, f): sample a random "lock" α, then run the SKFE secret-key encryption for a function $H_{\alpha,f}$ which computes the following multiple-output-bit functionality

$$H_{\alpha,f}(x) = \begin{cases} \alpha & \text{if } f(x) = 0 \\ 0 & \text{else} \end{cases}.$$

  Then, produce the lockable obfuscation $\mathsf{Obf}[P_{\mathsf{FE.ct}_H}, \alpha]$ as the ciphertext, where $P_{\mathsf{FE.ct}_H}(Y)$ parses Y as a SKFE secret key and computes the SKFE decryption functionality.
- PK-Enc: Use the simulator of lockable obfuscation to get a dummy obfuscated program of appropriate size. The program outputs "$\perp$" on every input.
- Dec: Run the obfuscated program, if it outputs "$\perp$" then output 1, else output 0.

We need mixed-FE to satisfy two security conditions. First, an adversary's view given polynomially many secret keys for inputs x, and at most t (secret key) ciphertexts for functions $f_1, \ldots, f_t$ can be simulated given only the function evaluations $f_i(x)$ for all x. This property, called functional indistinguishability, follows directly from the security of the SKFE. Indeed, we do not rely on obfuscation security here. The second security property, called secret/public mode indistinguishability, says that a public-key encryption and a secret key encryption of the trivial branching program f (which outputs 1 on all inputs) are computationally indistinguishable. Furthermore, this should hold even given polynomially many SKFE keys and $t-1$ SKFE ciphertexts for arbitrary functions. This property follows from a combination of symmetric-FE security and lockable obfuscation, by first changing the symmetric-FE ciphertext from $H_{\alpha,f}$ to the "all $\perp$" function, then changing real obfuscation to simulated using the lockable obfuscation security.

1.3 Mixed FE from Private Constrained PRFs

A constrained PRF is a standard PRF with the additional property that given a program M and a PRF key K, we can create a constrained key that allows someone to evaluate the PRF at inputs x where $M(x) = 0$ while randomizing the outputs of all other inputs. A private constrained PRF (PCPRF) satisfies the additional requirement that the constrained key hides M (in the appropriate sense). We will work with a strengthening of this requirement, which says that given M along with a sequence of inputs $\{x_i\}$ such that $M(x_i) = 1$, the joint distribution of the constrained key for M along with the PRF evaluations at $\{x_i\}$ are pseudorandom.

We show how to construct a 1-CT mixed-FE scheme starting from any PCPRF. We then show how to "boost" this basic construction to a t-CT mixed-FE, assuming that the underlying PCPRF is also key-homomorphic [5], namely for all K, K', x, we have $\mathsf{PRF}_{K+K'}(x) \approx \mathsf{PRF}_K(x) + \mathsf{PRF}_{K'}(x)$. Our schemes achieve simulation-based security, and support functions computable by polynomial-size circuits.

1-CT Scheme. We observe that a PCPRF scheme already gives a 1-CT mixed FE scheme:

- Setup: Choose a master secret key msk for the PCPRF.
- SKGen(msk, x): A secret key for x is a PRF evaluation at x;
- SK-Enc(msk, M): An encryption of a program M is a constrained key for M;
- PK-Enc: Use the simulator of the PCPRF to produce a simulated constrained key.
- Dec: To decrypt, we compare the constrained evaluation at x with the PRF evaluation at x; if they are equal, we output 0, and otherwise, we output 1.

The existing security proofs show that, if for all x_i we have $M(x_i) = 1$, then the constrained key for the program is computationally indistinguishable from a random key that is independent of the PRF evaluations. This means that we can *obliviously* sample encryptions of the "always-1" program by sampling a random ciphertext.

From 1-CT to 2-CT. We provide an almost generic transformation from a 1-CT to a 2-CT scheme, assuming that the underlying scheme is key-homomorphic, and also satisfies a natural distribution requirement. Namely, we require that for $\mathsf{msk}_1, \mathsf{msk}_2, \mathsf{msk}'$ that are correctly generated from the 1-CT mixed FE scheme, the distributions of $\mathsf{msk}_1 + \mathsf{msk}_2$, $\mathsf{msk}_1 - \mathsf{msk}_2$, and msk' are identical. In addition, for all x, we have

$$\mathsf{skGen}(\mathsf{msk}_1, x) + \mathsf{skGen}(\mathsf{msk}_2, x) = \mathsf{skGen}(\mathsf{msk}_1 + \mathsf{msk}_2, x)$$

When the 1-CT mixed FE schemes are instantiated by the PCPRFs in [11–13], they satisfy an approximate notion of key-homomorphism, which suffices for the purpose of constructing collusion resistant mixed FE. In the rest of the

introduction we assume the underlying PCPRFs are exact key-homomorphic for simplicity, and leave the instantiations from the approximate ones in the main body.

Our 2-CT mixed FE scheme works as follows:

- Setup: choose λ pairs of $\mathsf{msk}_{i,b}$ as the master secret keys for the 1-CT scheme;
- SKGen(msk, x): The secret key for x runs the secret-key generation algorithm for the 1-CT scheme over all the λ pairs of $\mathsf{msk}_{i,b}$, outputs $\{\mathsf{skGen}(\mathsf{msk}_{i,b}, x)\}_{i\in[\lambda],b\in\{0,1\}}$;
- SK-Enc(msk, M): To encrypt a program M, we pick a random $\mathbf{z} \in \{0,1\}^\lambda$, output $\mathbf{z}$ and the 1-CT encryption SK-Enc($\mathsf{msk}_{\mathbf{z}}, M$) as the ciphertext, where $\mathsf{msk}_{\mathbf{z}} := \mathsf{msk}_{1,z_1} + \cdots + \mathsf{msk}_{\lambda,z_\lambda}$;
- PK-Enc: pick a random $\mathbf{z} \in \{0,1\}^\lambda$, then run the PK-Enc mode of the 1-CT scheme.
- Dec: To decrypt, first derive $\mathsf{skGen}(\mathsf{msk}_{\mathbf{z}}, x) = \sum_{i=1}^{\lambda} \mathsf{skGen}(\mathsf{msk}_{i,z_i}, x)$ and then run the 1-CT decryption algorithm.

Next, we sketch a proof of security by constructing a simulator for the 2-CT scheme, starting from that for the 1-CT scheme. Suppose we want to simulate encryptions of two programs M^1, M^2 under tags $\mathbf{z}^1, \mathbf{z}^2$. The only property we need from $\mathbf{z}^1, \mathbf{z}^2$ is that they differ in one bit position, which happens with probability $1 - 2^{-\lambda}$. For notational simplicity, assume that

$$\mathbf{z}^1 = 00\cdots0, \mathbf{z}^2 = 10\cdots0$$

Now, using the simulator for the 1-CT scheme (and a hybrid argument), we can simulate the 1-CT encryptions SK-Enc$(\widetilde{\mathsf{msk}}_1, M^1)$, SK-Enc$(\widetilde{\mathsf{msk}}_2, M^2)$ for two random $\widetilde{\mathsf{msk}}_1, \widetilde{\mathsf{msk}}_2$, along with $\mathsf{skGen}(\widetilde{\mathsf{msk}}_1, x)$ and $\mathsf{skGen}(\widetilde{\mathsf{msk}}_2, x)$ for arbitrarily many x's.

To construct a simulator for the 2-CT scheme, we follow the natural simulation strategy where we pick $\mathsf{msk}_{i,b}$ and program

$$\mathsf{msk}_{\mathbf{z}^1} = \widetilde{\mathsf{msk}}_1, \mathsf{msk}_{\mathbf{z}^2} = \widetilde{\mathsf{msk}}_2$$

as follows:

- We sample $(\mathsf{msk}_{i,0}, \mathsf{msk}_{i,1}), i = 2, \ldots, \lambda$ ourselves;
- We implicitly program

$$\mathsf{msk}_{1,0} = \widetilde{\mathsf{msk}}_1 - \sum_{i=2}^{\lambda} \mathsf{msk}_{i,0}, \mathsf{msk}_{1,1} = \widetilde{\mathsf{msk}}_2 - \sum_{i=2}^{\lambda} \mathsf{msk}_{i,0}$$

Simulating the ciphertexts is straight-forward. To simulate a key $\{\mathsf{skGen}(\mathsf{msk}_{i,b}, x)\}$ for x,

- We can compute $\mathsf{skGen}(\mathsf{msk}_{i,0}, x), \mathsf{skGen}(\mathsf{msk}_{i,1}, x), i = 2, \ldots, \lambda$ ourselves since we know $\mathsf{msk}_{i,0}, \mathsf{msk}_{i,1}$;

– We can compute $\mathsf{skGen}(\mathsf{msk}_{1,0}, x)$ using the key-homomorphic property via

$$\mathsf{skGen}(\mathsf{msk}_{1,0}, x) = \mathsf{skGen}(\widetilde{\mathsf{msk}}_1, x) - \sum_{i=2}^{\lambda} \mathsf{skGen}(\mathsf{msk}_{i,0}, x)$$

We can similarly compute $\mathsf{skGen}(\mathsf{msk}_{1,1}, x)$.

From 2-CT to t-CT. To obtain a scheme that is secure for t ciphertexts, we follow [20, Remark 8.1] and sample each entry of the tag $\mathbf{z}$ from a larger alphabet. The natural extension of the previous argument is to require that with high probability over $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(t)}$, there exists $j^* \in [\lambda]$ such that $z^{(1)}_{j^*}, \ldots, z^{(t)}_{j^*}$ are all distinct. This would require an alphabet of size $\Omega(t^2)$. Instead, we observe that it suffices that there exists $j^*_1, \ldots, j^*_t \in [\lambda]$ such that the t pairs $(j^*_1, z^{(1)}_{j^*_1}), \ldots, (j^*_t, z^{(t)}_{j^*_t})$ are distinct (the natural extension corresponds to the special case $j^*_1 = \cdots = j^*_t = j^*$); this relaxation allows us to work with an alphabet of size $O(t)$. In the security proof, we will receive ciphertexts and secret keys corresponding to t independent $\widetilde{\mathsf{msk}}_1, \ldots, \widetilde{\mathsf{msk}}_t$, which we "embed" into $\mathsf{msk}_{j^*_1, z^{(1)}_{j^*_1}}, \ldots, \mathsf{msk}_{j^*_t, z^{(t)}_{j^*_t}}$.

We proceed to describe our construction in a bit more detail. We replace λ pairs of master secret keys $\{\mathsf{msk}_{j,d}\}_{j\in[\lambda], d\in\{0,1\}}$ (in the 2-CT scheme) with λ many $(2t-2)$-tuples of $\{\mathsf{msk}_{j,d}\}_{j\in[\lambda], d\in[2t-2]}$, and sample the tag $\mathbf{z}$ from $[2t-2]^\lambda$. For each tag $\mathbf{z}^{(i)}$, $i \in [t]$, the probability that the j^{th} coordinate of $\mathbf{z}^{(i)}$ does not show up in the other $t-1$ tags is $\geq \frac{(2t-2)-(t-1)}{2t-2} = \frac{1}{2}$, therefore the probability that one of the coordinate of $\mathbf{z}^{(i)}$ is unique is at least $1 - 2^{-\lambda}$ (this unique coordinate corresponds to j^*_i). By a union bound, with probability at least $1 - t \cdot 2^{-\lambda}$, all the tags has one unique coordinate.

1.4 Attribute-Based Traitor Tracing

Finally, we very briefly describe our results on attribute-based traitor tracing. An attribute-based traitor-tracing (AB-TT) scheme is like an ABE with tracing capabilities. The key generation algorithm gives out secret keys $\mathsf{sk}_{f,i}$ for functions f with respect to some identity i. The encryption procedure encrypts a message m with respect to an attribute x and the resulting ciphertext can be correctly decrypted by $\mathsf{sk}_{f,i}$ if $f(x) = 1$. The identity i is completely irrelevant from the point of view of ABE correctness/security. The tracing algorithm is given a decoder D which is able to distinguish between the encryptions of some two messages m_0, m_1 with respect to some attribute x. The goal is to recover some traitor i whose key $\mathsf{sk}_{f,i}$ was used in the creation of the decoder *and* who is qualified to decrypt meaning that $f(x) = 1$. Note that there may be many other traitors that participate in the creation of the decoder and who are not qualified to decrypt (e.g., have keys $\mathsf{sk}_{g,j}$ for some g such that $g(x) = 0$) but the tracing algorithm must find a traitor who is qualified to decrypt.

We argue that catching a qualified user is the correct definition for tracing. For example, imagine that a system is setup such that for a certain attribute x corresponds to extremely sensitive information that only highly positioned individuals can access. By the ABE security properties, if a decoder D were discovered that could decrypt such ciphertexts it must be the case that such a highly positioned user contributed to it. It would be rather unsatisfying if a tracing algorithm were only able to finger a lower level individual that contributed to it. We note that such tracing definitions were considered in prior works [1,21–23], however, any black box tracing in such works required a $\sqrt{n}$ factor of ciphertext blowup for n users which was inherited from [7,8]. We improve this to $\mathsf{polylog}(n)$, by constructing AB-TT from attribute-based mixed FE, which can be obtained from ABE and mixed-FE for all polynomial-time computations. For more details, we refer the reader to Sect. 5.

Additional Related Work on Tracing. Our work and comparisons focus on tracing schemes that are collusion resistant. Starting with [14] there existed many cryptosystems that would be collusion resistant up to t corrupted users where t was some parameter of system setup. See [3] and the references therein for further discussion of collusion bounded systems. Boneh, Sahai and Waters [7] gave the first collusion resistant tracing schemes with ciphertext size that was sublinear in the number of users n. They achieved ciphertext growth proportional to $\sqrt{n}$ using composite order bilinear groups. Later variants [8,15,16] achieved similar ciphertext size under improved bilinear assumptions. Several years later Boneh and Zhandry [9] utilized indistinguishability obfuscation to achieve the ideal case where ciphertexts grow polynomially in $\log(n)$ and λ. However, indistinguishability obfuscation is not known from standard assumptions.

2 Preliminaries

Notations and Terminology. In cryptography, the security parameter (denoted as λ) is a variable that is used to parameterize the computational complexity of the cryptographic algorithm or protocol, and the adversary's probability of breaking security. An algorithm is "efficient" if it runs in (probabilistic) polynomial time over λ.

When a variable v is drawn randomly from the set S we denote as $v \xleftarrow{\$} S$ or $v \leftarrow U(S)$, sometimes abbreviated as v when the context is clear. We use $\approx_s$ and $\approx_c$ as the abbreviation for statistically close and computationally indistinguishable.

Let $\mathbb{R}, \mathbb{Z}, \mathbb{N}$ be the set of real numbers, integers and positive integers. Denote $\mathbb{Z}/(q\mathbb{Z})$ by $\mathbb{Z}_q$. For $n \in \mathbb{N}$, $[n] := \{1, ..., n\}$. A vector in $\mathbb{R}^n$ (represented in column form by default) is written as a bold lower-case letter, e.g. $\mathbf{v}$. For a vector $\mathbf{v}$, the i^{th} component of $\mathbf{v}$ will be denoted by v_i. A matrix is written as a bold capital letter, e.g. $\mathbf{A}$. The i^{th} column vector of $\mathbf{A}$ is denoted $\mathbf{a}_i$. The length of a vector is the ℓ_p-norm $\|\mathbf{v}\|_p = (\sum v_i^p)^{1/p}$. The length of a matrix is the norm of its longest column: $\|\mathbf{A}\|_p = \max_i \|\mathbf{a}_i\|_p$. By default we use ℓ_2-norm unless explicitly mentioned. When a vector or matrix is called "small", we refer to its norm.

2.1 Learning with Errors

We recall the learning with errors problem.

Definition 2.1 (Decisional learning with errors (LWE) [26]). *For $n, m \in \mathbb{N}$ and modulus $q \geq 2$, distributions for secret vectors, public matrices, and error vectors $\theta, \pi, \chi \subseteq \mathbb{Z}_q$. An LWE sample is obtained from sampling $\mathbf{s} \leftarrow \theta^n$, $\mathbf{A} \leftarrow \pi^{n \times m}$, $\mathbf{e} \leftarrow \chi^m$, and outputting $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \mod q)$.*

We say that an algorithm solves $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$ if it distinguishes the LWE sample from a random sample distributed as $\pi^{n \times m} \times U(\mathbb{Z}_q^{1 \times m})$ with probability bigger than $1/2$ plus non-negligible.

Lemma 2.2 (Standard form [10,24–26]). *Given $n \in \mathbb{N}$, for any $m = \mathrm{poly}(n)$, $q \leq 2^{\mathrm{poly}(n)}$. Let $\theta = \pi = U(\mathbb{Z}_q)$, $\chi = D_{\mathbb{Z},\sigma}$ where $\sigma \geq 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that breaks $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$, then there exists an efficient (possibly quantum) algorithm for approximating SIVP and GapSVP in the ℓ_2 norm, in the worst case, to within $\tilde{O}(nq/\sigma)$ factors.*

We drop the subscripts of LWE when referring to standard form of LWE with the parameters specified in Lemma 2.2.

2.2 Secret-Key and Mixed Functional Encryption

t-CT SKFE. We begin with the definition for SKFE:

Definition 2.3 (Secret-key functional encryption (SKFE)). *A secret-key functional encryption scheme for a class of functions $\mathcal{F}_\mu = \{f : \{0,1\}^\mu \to \{0,1\}\}$ is a tuple of probabilistic polynomial time (p.p.t) algorithms $(\mathsf{Setup}, \mathsf{skGen}, \mathsf{skEnc}, \mathsf{Dec})$ such that:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter 1^λ, and outputs the master secret key msk and the public parameters pp.*
- $\mathsf{skGen}(\mathsf{msk}, m)$ *takes as input msk and a message $m \in \{0,1\}^\mu$, and outputs a decryption key sk_m.*
- $\mathsf{skEnc}(\mathsf{msk}, f)$ *takes as input msk and a function $f \in \mathcal{F}_\mu$, and outputs a ciphertext ct.*
- $\mathsf{Dec}(\mathsf{sk}_m, \mathsf{ct})$ *takes as input sk_m and ct, and outputs a single bit.*

Correctness. For every message $m \in \{0,1\}^\mu$ and function $f \in \mathcal{F}_\mu$ we have:

$$\Pr[\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda);\ \mathsf{sk}_m \leftarrow \mathsf{skGen}(\mathsf{msk}, m) : \mathsf{Dec}(\mathsf{sk}_m, \mathsf{skEnc}(\mathsf{msk}, f)) = f(m)] = 1 - \mathrm{negl}(\lambda),$$

where the probability is taken over the randomness of the algorithms $\mathsf{Setup}, \mathsf{skGen}, \mathsf{skEnc}, \mathsf{Dec}$.

Function-Hiding Security. For all p.p.t stateful algorithms Adv, there is a p.p.t. stateful algorithm Sim such that:

$$\left\{\text{Experiment } \mathrm{REAL}_{\mathsf{Adv}}(1^\lambda)\right\}_{\lambda \in \mathbb{N}} \approx_c \left\{\text{Experiment } \mathrm{IDEAL}_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)\right\}_{\lambda \in \mathbb{N}}$$

where the real and ideal experiments of stateful algorithms $\mathsf{Adv}, \mathsf{Sim}$ are as follows:

Experiment $\mathrm{REAL}_{\mathsf{Adv}}(1^\lambda)$	Experiment $\mathrm{IDEAL}_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)$
$\mathsf{msk} \leftarrow \mathsf{Gen}(1^\lambda)$,	$\mathsf{Sim} \leftarrow 1^\lambda$
For $i \in [t]$:	For $i \in [t]$:
$\mathsf{Adv} \rightarrow f^{[i]}$;	$\mathsf{Adv} \rightarrow f^{[i]}$;
$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i]} = \mathsf{skEnc}(\mathsf{pp}, \mathsf{msk}, f^{[i]})$;	$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i]} = \mathsf{Sim}(1^{\lvert f^{[i]} \rvert})$;
Repeat polynomially many times:	Repeat polynomially many times:
$\mathsf{Adv} \rightarrow m$; $\mathsf{Adv} \leftarrow \mathsf{skGen}(\mathsf{pp}, \mathsf{msk}, m)$	$\mathsf{Adv} \rightarrow m$; $\mathsf{Adv} \leftarrow \mathsf{Sim}(m, \left\{ f^{[i]}(m) \right\}_{i \in [t]})$
$\mathsf{Adv} \rightarrow b$; Output b	$\mathsf{Adv} \rightarrow b$; Output b

In the experiments, the adversary Adv can ask for t ciphertexts followed by polynomially many decryption key queries. Once Adv makes a ciphertext query for a function $f \in \mathcal{F}_\lambda$, in the real experiment Adv obtains the ciphertext generated by the secret-key encryption algorithm; in the ideal experiment Adv obtains the ciphertext generated by Sim given only the (circuit) size of f. Once Adv makes a message query m, in the real experiment Adv obtains sk_m from the decryption key generation algorithm; in the ideal experiment, Adv obtains the decryption key generated by the simulator who is given m, and $\left\{ f^{[i]}(m) \right\}_{i \in [t]}$. The output of the experiment is the final output bit of Adv.

Remark 2.4 (adaptive security). A t-CT SKFE scheme is called adaptively secure if the function and ciphertext queries can be made adaptively in any order. Some constructions achieve partially adaptive security and we will explicitly mention the restrictions.

t-CT Mixed FE. We provide a simulation-based definition for t-ciphertext (t-CT) mixed-FE, which is same as the definition in [20, Sect. 5] where it is referred to as $(t-1)$-bounded mixed-FE.

Definition 2.5 (Mixed functional encryption). *A mixed functional encryption scheme for a class of functions $\mathcal{F}_\mu = \{f : \{0,1\}^\mu \rightarrow \{0,1\}\}$ is a tuple of probabilistic polynomial time (p.p.t) algorithms* $(\mathsf{Setup}, \mathsf{skGen}, \mathsf{skEnc}, \mathsf{Dec}, \mathsf{pkEnc})$ *such that:*

- $(\mathsf{Setup}, \mathsf{skGen}, \mathsf{skEnc}, \mathsf{Dec})$ *are the same as SKFE.*
- $\mathsf{pkEnc}(\mathsf{pp})$ *takes as input* pp*, and outputs a ciphertext* ct.

Correctness and Function-Hiding Security. *Same as SKFE.*

Public/Secret-Key Mode Indistinguishability. *In addition to the security requirement above for a normal secret-key functional encryption, a mixed-FE further requires that for a function f queried to the encryption oracle, if for all message m queried by the adversary, $f(m) = 1$ (the other potential $t-1$ functions does not have to satisfy this requirement), then the secret-key ciphertext* $\mathsf{skEnc}(\mathsf{msk}, f)$ *is indistinguishable from a sample from* $\mathsf{pkEnc}(\mathsf{pp})$*. Formally, we*

require that for all p.p.t stateful algorithms Adv*, the following two experiments produce indistinguishable outputs:*

$$\left\{\textit{Experiment } SKEXP_{\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\textit{Experiment } PKEXP_{\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

The experiments are as follows:

Experiment $SKEXP_{\mathsf{Adv}}(1^\lambda)$	*Experiment* $PKEXP_{\mathsf{Adv}}(1^\lambda)$
$\mathsf{pp}, \mathsf{msk} \leftarrow \mathsf{Gen}(1^\lambda)$,	$\mathsf{pp}, \mathsf{msk} \leftarrow \mathsf{Gen}(1^\lambda)$,
For i *in* $[i^* - 1]$*:*	*For* i *in* $[i^* - 1]$*:*
$\mathsf{Adv} \rightarrow f^{[i]}$;	$\mathsf{Adv} \rightarrow f^{[i]}$;
$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i]} = \mathsf{skEnc}(\mathsf{msk}, f^{[i]})$;	$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i]} = \mathsf{skEnc}(\mathsf{msk}, f^{[i]})$;
$\mathsf{Adv} \rightarrow f^{[i^*]}$;	$\mathsf{Adv} \rightarrow f^{[i^*]}$;
$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i^*]} = \mathsf{skEnc}(\mathsf{msk}, f^{[i^*]})$;	$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i^*]} = \mathsf{pkEnc}(\mathsf{pp})$;
For i *in* $[i^* + 1, t]$*:*	*For* i *in* $[i^* + 1, t]$*:*
$\mathsf{Adv} \rightarrow f^{[i]}$;	$\mathsf{Adv} \rightarrow f^{[i]}$;
$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i]} = \mathsf{skEnc}(\mathsf{msk}, f^{[i]})$;	$\mathsf{Adv} \leftarrow \mathsf{ct}^{[i]} = \mathsf{skEnc}(\mathsf{msk}, f^{[i]})$;
Repeat polynomially many times:	*Repeat polynomially many times:*
$\mathsf{Adv} \rightarrow m$; $\mathsf{Adv} \leftarrow \mathsf{skGen}(\mathsf{msk}, m)$	$\mathsf{Adv} \rightarrow m$; $\mathsf{Adv} \leftarrow \mathsf{skGen}(\mathsf{msk}, m)$
$\mathsf{Adv} \rightarrow b$; *Output* b	$\mathsf{Adv} \rightarrow b$; *Output* b

Remark 2.6 (comparison with [20]). What we call a t-CT mixed FE is referred as a $(t-1)$-query mixed FE in [20]. In the latter, security is formalized using a indistinguishability-based paradigm. For the public/secret-key mode indistinguishability, they require that the two ciphertexts are indistinguishable given honestly generated secret keys from skGen.

3 t-CT Mixed-FE from Lockable Obfuscation and t-CT SKFE

In this section, we present a construction of t-ciphertext mixed-FE for the class of all poly-time computable functions from any lockable obfuscation for all poly-time computable functions and t-ciphertext secret key functional encryption for all poly-time computable functions. Thus, our construction shows how to use lockable obfuscation to generically add a public-key oblivious sampling mode to a SKFE.

3.1 Lockable Obfuscation

Recall the definition of lockable obfuscation from [19,28].

Definition 3.1 (Lockable (or compute-and-compare) obfuscation). *Consider a family of functions* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *where* $\mathcal{F}_\lambda = \left\{f : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\nu(\lambda)}\right\}$, $\nu(\lambda) = \omega(\log\lambda)$. *A lockable obfuscator takes a function* $f \in \mathcal{F}$ *and a target* $\alpha \in \{0,1\}^\nu$, *outputs an obfuscated program* $\mathsf{Obf}[f,\alpha]$ *which satisfies the following properties:*

Functionality. $\mathsf{Obf}[f,\alpha]$ *takes an input* $x \in \{0,1\}^\ell$, *output* 1 *if* $f(x) = \alpha$; $\perp$ *otherwise.*

Virtual Black-Box Security. A lockable obfuscator is said to satisfy virtual black-box security if there is a p.p.t. simulator S *such that for all* $f \in \mathcal{F}$,

$$\mathsf{Obf}[f,\alpha] \approx_c S(1^\lambda, 1^{|f|})$$

over $\alpha \xleftarrow{\$} \{0,1\}^\nu$ *and the randomness of the obfuscator and* S.

3.2 The Mixed-FE Construction

Construction 3.2. *Given a t-CT SKFE* $\mathsf{FE} = (\mathsf{FE.Gen}, \mathsf{FE.skGen}, \mathsf{FE.skEnc}, \mathsf{FE.Dec})$ *and a lockable obfuscator* Obf, *construct a t-CT mixed-FE as follows.*

- $\mathsf{Setup}(1^\lambda)$ *runs* $\mathsf{FE.msk} \leftarrow \mathsf{FE.Gen}(1^\lambda)$, *and treat it as the master secret key.*
- $\mathsf{skGen}(\mathsf{msk}, x)$ *outputs* $\mathsf{FE.sk}_x \leftarrow \mathsf{FE.skGen}(\mathsf{FE.msk}, x)$.
- $\mathsf{pkEnc}(\mathsf{pp})$ *outputs the simulated code for the lockable obfuscation* $\mathsf{Obf}.S(1^\lambda, 1^{\mathrm{poly}(|f|)})$.
- $\mathsf{skEnc}(\mathsf{msk}, f)$ *samples a random string* $\alpha \leftarrow \{0,1\}^\lambda$, *runs* $\mathsf{FE.ct}_H \leftarrow \mathsf{FE.skEnc}(\mathsf{msk}, H_{\alpha,f})$ *where* $H_{\alpha,f}$ *computes the following multiple-output-bit functionality*

$$H_{\alpha,f}(x) = \begin{cases} \alpha & \text{if } f(x) = 0 \\ 0 & \text{else} \end{cases}.$$

  Then, produce the lockable obfuscation $\mathsf{Obf}[P_{\mathsf{FE.ct}_H}, \alpha]$ *as the ciphertext, where* $P_{\mathsf{FE.ct}_H}(Y)$ *computes* $\mathsf{FE.Dec}(\mathsf{FE.ct}_H, Y)$.
- $\mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct})$ *parses* sk_x *as* $\mathsf{FE.sk}_x$, *and* ct *as* $\mathsf{Obf}[P_{\mathsf{FE.ct}_H}, \alpha]$, *outputs* $\mathsf{Obf}[P_{\mathsf{FE.ct}_H}, \alpha](\mathsf{FE.sk}_x)$.

3.3 The Security Analysis

Theorem 3.3. *Construction 3.2 is a t-CT mixed-FE assuming the underlying obfuscator* Obf *is a lockable obfuscation and* FE *is a t-CT secure secret-key FE.*

The only additional property (compared to a normal SKFE) is the indistinguishability of the public-key and the secret-key ciphertext for a function f s.t. for all x queried in the game, $f(x) = 1$. The intuition is that in that case, the α in the SKFE ciphertexts is hidden following the plain SKFE security. Therefore the α in the lockable obfuscation target is random and independent, and we can trigger the simulation security of the lockable obfuscation.

Proof. We prove the indistinguishability of the public and secret-key modes.

Consider the following intermediate distribution for the t-CT mixed-FE experiment. Once Adv makes a ciphertext query for a function $f^{(i)} \in \mathcal{F}_\lambda$, $i \in [t]$, the challenger responds by sampling a random string $\alpha^{(i)} \leftarrow \{0,1\}^\lambda$, runs $\mathsf{FE.ct}_H \leftarrow \mathsf{FE.Sim}(1^{\mathrm{poly}\,|H_{f^{(i)}}|})$. Then produces the lockable obfuscation $\mathsf{Obf}[P_{\mathsf{FE.ct}_H}, \alpha^{(i)}]$ as the ciphertext, where $P_{\mathsf{FE.ct}_H}(Y)$ computes $\mathsf{FE.Dec}(\mathsf{FE.ct}_H, Y)$. Once Adv makes a message query m, the challenger responds with a decryption key $\mathsf{FE.sk}_m \leftarrow \mathsf{FE.Sim}(m, \{1^{(i)}\}_{i\in[t]})$.

So if there is an adversary that distinguishes the real distribution and the intermediate distribution, then there is an adversary that breaks the simulation security for the t-CT SKFE. If there is an adversary that distinguishes the intermediate distribution from the public key mode, then we build an adversary that breaks the lockable obfuscation, due to the fact that the $P_{\mathsf{FE.ct}_H}$ in the intermediate distribution does not depend on $\alpha^{(i)}$.

4 *t*-CT Mixed-FE from Key-Homomorphic Private Constrained PRF

In this section we present a construction of mixed-FE from key-homomorphic PCPRF.

4.1 Background of Key-Homomorphic Private Constrained PRFs

We first give the definition of a key-homomorphic private constrained PRF, which literally combines key-homomorphism [5] with private constrained PRFs [6,12]. For the purpose of this paper we work with the KHPCPRFs that satisfy the simulation-based definition given one constrained key and many input queries. We then explain that the PCPRF constructions in [11–13] satisfy an approximate version of key-homomorphism, which suffices for constructing mixed-FE.

Definition 4.1 (Key-homomorphic private constrained PRF (KHPCPRF)). *Consider a family of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{F}_\lambda = \{F_k : D_\lambda \to R_\lambda\}$, along with a tuple of efficient functions (*ppGen, skGen, $\mathsf{Constrain}$, Eval, $\mathsf{Constrain.Eval}$*). For a constraint family $\mathcal{C} = \{\mathcal{C}_\lambda = \{C : D_\lambda \to \{0,1\}\}\}_{\lambda\in\mathbb{N}}$,*

- *The public parameter generation algorithm* $\mathsf{ppGen}(1^\lambda, \mathcal{F}_\lambda)$ *takes the security parameter λ and the description of the constraint class $\mathcal{F}_\lambda$, generates the public parameter* pp.
- *The secret key generation algorithm* $\mathsf{skGen}(1^\lambda, \mathsf{pp})$ *takes the security parameter λ, and the public parameter* pp*, generates the secret key* sk.
- *The evaluation algorithm* $\mathsf{Eval}(\mathsf{sk}, x)$ *takes* sk*, an input x, outputs $F_{\mathsf{sk}}(x)$.*
- *The constraining algorithm* $\mathsf{Constrain}(1^\lambda, \mathsf{pp}, \mathsf{sk}, C)$ *takes* sk*, a constraint $C \in \mathcal{C}_\lambda$, outputs the constrained key* ck_C.

– *The constrained evaluation algorithm* $\mathsf{Constrain.Eval}(\mathsf{ck}_C, x)$ *takes a constrained key* ck_C*, an input* x*, outputs* $F_{\mathsf{ck}_C}(x)$*.*

$\mathcal{F}$ *is called a family of* key-homomorphic private constrained PRF *for* $\mathcal{C}$ *if it satisfies the following properties:*

Functionality preservation for $C(x) = 0$*. For any constraint* $C \in \mathcal{C}_\lambda$*, any input* $x \in D_\lambda$ *s.t.* $C(x) = 0$*,*

$$\Pr[\mathsf{Eval}(\mathsf{sk}, x) = \mathsf{Constrain.Eval}(\mathsf{ck}_C, x)] \geq 1 - \mathrm{negl}(\lambda),$$

where the probability is taken over the randomness in algorithms ppGen*,* skGen *and* Constrain*.*

Pseudorandomness and Constraint-Hiding. For any polynomial time algorithm Adv*, there is a polynomial time algorithm* Sim *such that:*

$$\left\{\textit{Experiment } REAL_{\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\textit{Experiment } IDEAL_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

where the ideal and real experiments are defined as follows. In the experiments the adversary can ask a single constraint query followed by polynomially many input queries. Once Adv *makes the constraint query* $C \in \mathcal{C}_\lambda$*, in the real experiment* Adv *obtains the constrained key generated by the constraining algorithm; in the ideal experiment* Adv *obtains a key generated by* Sim*, whereas* Sim *is given only the size of* C*. Once* Adv *makes an input query* x*,* Adv *is expected to provide a bit* d_x *indicating the value of* $C(x)$*. In the real experiment* Adv *obtains the unconstrained function value at* x*. In the ideal experiment* Sim *learns the indicator bit* d_x*; if* $d_x = 0$ *then* Adv *gets a value generated by* Sim*, and if* $d_x = 1$ *then* Adv *obtains a random value from the range* R *of the function. The output of the experiment is the final output bit of* Adv*.*

Experiment $REAL_{\mathsf{Adv}}(1^\lambda)$	*Experiment* $IDEAL_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{ppGen}(1^\lambda)$,	$\mathsf{Sim} \leftarrow 1^\lambda$
$\mathsf{sk} \leftarrow \mathsf{skGen}(1^\lambda, \mathsf{pp})$,	$\mathsf{Sim} \leftarrow 1^\lambda$
$\mathsf{Adv} \rightarrow C$;	$\mathsf{Adv} \rightarrow C$;
$\mathsf{Adv} \leftarrow \mathsf{Constrain}(\mathsf{pp}, \mathsf{sk}, C)$	$\mathsf{Adv} \leftarrow \mathsf{Sim}(1^{\lvert C\rvert})$
Repeat :	*Repeat* :
$\mathsf{Adv} \rightarrow x$; $y = \mathsf{Eval}(\mathsf{sk}, x)$	$\mathsf{Adv} \rightarrow x$; $y = \mathsf{Sim}(x, d_x)$
$\mathsf{Adv} \leftarrow y$	*if* $d_x = 1$ *then* $y = U(R)$; $\mathsf{Adv} \leftarrow y$
$\mathsf{Adv} \rightarrow b$; *Output* b	$\mathsf{Adv} \rightarrow b$; *Output* b

Key-Homomorphism for the SK. Let $\circ$ *denote the group operation. For* $\mathsf{pp} \leftarrow \mathsf{ppGen}(1^\lambda, \mathcal{F}_\lambda)$*,* $\mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{skGen}(1^\lambda, \mathsf{pp})$*, and any input* $x \in D_\lambda$*.*

$$\Pr[\mathsf{Eval}(\mathsf{sk}_1 \circ \mathsf{sk}_2, x) = \mathsf{Eval}(\mathsf{sk}_1, x) \circ \mathsf{Eval}(\mathsf{sk}_2, x)] \geq 1 - \mathrm{negl}(\lambda).$$

The Distribution Requirement on the Secret Keys. Let $\circ$ *denote the group operation. We additionally require that for* $\mathsf{pp} \leftarrow \mathsf{ppGen}(1^\lambda, \mathcal{F}_\lambda)$*, for* $\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}'$ *sampled from* $\mathsf{skGen}(1^\lambda, \mathsf{pp})$ *with independent randomness,* $\mathsf{sk}_1 \circ \mathsf{sk}_2$*,* $\mathsf{sk}_1 \circ (-\mathsf{sk}_2)$*, and* sk' *are from the same distribution.*

Almost Key-Homomorphic PCPRF and the LWE-Based Constructions. The existing LWE-based PCPRFs satisfies the notion of almost-key-homomorphism. For simplicity we focus on the case where the range of the PRF is $\mathbb{Z}_p^n$ for $n, p \in \mathbb{N}$, and let the operation be $+$, which is what the LWE-based PCPRFs work with.

Definition 4.2 (Almost-Key-Homomorphism). *For* $B, p, n \in \mathbb{N}$ *such that* $B < p$*. Let* $+$ *be the group operation. A family of PRFs* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ *with domain* D_λ *and range* $\mathbb{Z}_p^n$ *is called* B*-almost-key-homomorphic if for* $\mathsf{pp} \leftarrow \mathsf{ppGen}(1^\lambda, \mathcal{F}_\lambda)$*,* $\mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{skGen}(1^\lambda, \mathsf{pp})$*, and any input* $x \in D_\lambda$*.*

$$\|\mathsf{Eval}(\mathsf{sk}_1, x) + \mathsf{Eval}(\mathsf{sk}_2, x) - \mathsf{Eval}(\mathsf{sk}_1 \circ \mathsf{sk}_2, x)\|_\infty \leq B$$

Next we briefly explain how to set the parameters for the existing lattice-based PCPRFs to achieve almost-key-homomorphism.

Let $q > p \geq 2$ be the moduli. In all the LWE-based PCPRF constructions, the evaluation algorithms first work entirely over $\mathbb{Z}_q^n$, then finalize by applying a (coordinate-wise) rounding operation $\lfloor a \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ by multiplying a by p/q and rounding the result to the nearest integer. For any $a, b \in \mathbb{Z}_q$, we have

$$|\lfloor a \rceil_p + \lfloor b \rceil_p - \lfloor a + b \rceil_p| \leq 1$$

For the two PCPRF constructions [12,13] for branching programs from the GGH15 approach [17]. Let h be the length of the branching program (i.e. the number of indexes), ℓ be the bit-length of the input, and π be the index-to-input mapping. Recall that the (secret-key) evaluation algorithm takes as input a sequence of matrices $\left\{\mathbf{S}_{i,b} \in Z_q^{n \times n}\right\}_{i \in [h], b}$ and a vector $\mathbf{a}$ sampled uniformly random from $\mathbb{Z}_q^n$, computes the output on $x \in \{0,1\}^\ell$ as

$$\mathbf{y} = \left\lfloor \prod_{i \in [h]} \mathbf{S}_{i, x_{\pi(i)}} \cdot \mathbf{a} \right\rceil_p$$

By treating $\mathbf{a}$ as the secret key, the matrices in $\left\{\mathbf{S}_{i,b} \in Z_q^{n \times n}\right\}_{i \in [h], b}$ as the public parameters (which is explicit proved in [13] and generalizable to the setting in [12]), we have 1-almost-key-homomorphism since for all $x \in \{0,1\}^\ell$,

$$\left\lfloor \prod_{i \in [h]} \mathbf{S}_{i, x_{\pi(i)}} \cdot \mathbf{a}_1 \right\rceil_p + \left\lfloor \prod_{i \in [h]} \mathbf{S}_{i, x_{\pi(i)}} \cdot \mathbf{a}_2 \right\rceil_p \in \left\lfloor \prod_{i \in [h]} \mathbf{S}_{i, x_{\pi(i)}} \cdot (\mathbf{a}_1 + \mathbf{a}_2) \right\rceil_p + \{-1, 0, 1\}$$

So setting p to be a bit larger than the appropriated number of key addition suffices for achieving a meaningful key-homomorphism. The distribution property holds since the secret key is sampled uniformly random from $\mathbb{Z}_q^n$.

For the PCPRF constructions for all poly-size circuits in [11], the first construction [11, Sect. 4] satisfies almost-key homomorphism and the distribution requirement. Very briefly, the construction uses a secret key $\mathbf{s}$ sampled uniformly random from $\mathbb{Z}_q^n$, and a set of matrices in the public parameter pp. The evaluator takes an input x and pp, derives a matrix $\mathsf{pp}(x)$ which is independent of the secret key, then computes $\left\lfloor \mathbf{s}^T \cdot \mathsf{pp}(x) \right\rceil_p$. The approximate-key homomorphism and the distribution property follow immediately.

Remark 4.3. One can also define key-homomorphism for the constrained keys in the natural way, although it is not used in this paper. Let us remark that the constrained keys of the PCPRFs from [12,13] are also key-homomorphic.

4.2 Constructing t-CT Mixed-FE from KHPCPRF

Next we construct a t-CT secure mixed-FE from key-homomorphic PCPRF. The construction achieves t-CT security with a $O(t)$ blow-up in the size of the functional decryption key, which is smaller than the other existing secret-key functional encryptions with bounded collusion.

We first describe the construction of a mixed-FE from a PCPRF with exact key-homomorphism, then explain how to modify the construction and security analysis slightly to work with the LWE-based almost-key-homomorphic PCPRFs.

Let us remark that the construction and security analysis for a 1-CT secure mixed-FE from PCPRF (even without key-homomorphism) is implicit in [12, Sect. 6] and is explained in the introduction. So we deal with the case of $2 \leq t \leq \mathrm{poly}(\lambda)$ directly.

Let $\mathcal{T} = \{1, 2, ..., 2t-2\}$. The idea is to pick $\lambda \times (2t-2)$ independently sampled secret keys for the KHPCPRF scheme, denote each of them as $\mathsf{sk}_{j,d}$, $j \in [\lambda]$, $d \in \mathcal{T}$. To generate a ciphertext for a function f, pick a random vector $\mathbf{z} \in \mathcal{T}^\lambda$, and encrypt the function f in the constrained key derived from the secret key $\sum_{j\in[\lambda]} \mathsf{sk}_{j,z_j}$. We then prove each encryption is a constrained key derived from an independently generated master secret key with overwhelming probability.

Construction 4.4. *Given a key-homomorphic PCPRF* F *with group operation* $+$*, domain* D *and range* R*, construct a* t*-CT secure mixed-FE* MFE *as follows.*

- $\mathsf{MFE.Setup}(1^\lambda)$ *runs* $\mathsf{F.ppGen}(1^\lambda)$ *to generate* $\mathsf{F.pp}$*. Then runs* $\mathsf{F.skGen}(1^\lambda, \mathsf{pp})$ *for* $\lambda \cdot (2t-2)$ *times with independent randomness, denote the resulting set of secret keys as* $\{\mathsf{F.sk}_{j,d}\}_{j\in[\lambda],d\in\mathcal{T}}$*. Let* $\mathsf{MFE.msk} := \mathsf{F.pp}, \{\mathsf{F.sk}_{j,d}\}_{j\in[\lambda],d\in\mathcal{T}}$*, let* $\mathsf{MFE.pp} := \mathsf{F.pp}$.
- $\mathsf{MFE.skGen}(\mathsf{MFE.msk}, x)$ *takes a message* $x \in D_\lambda$*, outputs*

$$\mathsf{sk}_x = x, \{\mathsf{F.Eval}(\mathsf{F.sk}_{j,d}, x)\}_{j\in[\lambda],d\in\mathcal{T}}.$$

- $\mathsf{MFE.skEnc}(\mathsf{MFE.msk}, f)$ *samples* $\mathbf{z} \xleftarrow{\$} \mathcal{T}^\lambda$*. Then let* $\mathsf{F.sk}_f = \sum_{j\in[\lambda]} \mathsf{F.sk}_{j,z_j}$*. Outputs*

$$\mathsf{ct}_f = \mathbf{z}, \mathsf{F.Constrain}(\mathsf{F.pp}, \mathsf{F.sk}_f, f).$$

– $\mathsf{MFE.pkEnc}(\mathsf{MFE.pp})$ *runs the simulator of* F *on the size of the maximum constraint* $1^{|C_\lambda|}$ *to generate a simulated constrained key* $\mathsf{F.Sim.ck}$*, outputs*

$$\mathsf{ct}_f = \mathbf{z} \xleftarrow{\$} \mathcal{T}^\lambda, \mathsf{F.Sim.ck}$$

– $\mathsf{MFE.Dec}(\mathsf{sk}_x, \mathsf{ct}_f)$ *parses* sk_x *as* $x, \{y_{j,d}\}_{j\in[\lambda],d\in\mathcal{T}}$ *and* ct_f *as* $\mathbf{z}, \mathsf{ck}_f$*, outputs*

$$\begin{cases} 0 & \text{if } \mathsf{F.Constrain.Eval}(\mathsf{ck}_f, x) = \sum_{j\in[\lambda]} y_{j,z_j} \\ 1 & \text{else} \end{cases}.$$

Correctness. For f and x such that $f(x) = 0$, then

$$\mathsf{F.Constrain.Eval}(\mathsf{ck}_f, x) = \mathsf{F.Eval}(\mathsf{F.sk}_f, x) = \sum_{j\in[\lambda]} \mathsf{F.Eval}(\mathsf{F.sk}_{j,z_j}, x),$$

where the first equality follows the correctness of the PCPRF, the second equality follows the exact key-homomorphism.

For f and x such that $f(x) = 1$, by the pseudorandomness of the PRF evaluations on x such that $f(x) = 1$, $\mathsf{F.Eval}(\mathsf{F.sk}_f, x)$ looks random, and is therefore unlikely to be equal to $\mathsf{F.Constrain.Eval}(\mathsf{ck}_f, x)$ as long as the range R is super-polynomially large.

Theorem 4.5. *Assuming* F *is a key-homomorphic PCPRF, Construction 4.4 gives a t-CT secure mixed-FE.*

Proof. We construct the mixed-FE simulator $\mathsf{MFE.Sim}(1^\lambda)$ as follows:

1. Preprocessing: Sample a set of tags $\left\{\mathbf{z}^{[i]} \xleftarrow{\$} \mathcal{T}^\lambda\right\}_{i\in[t]}$. We define $t+1$ sets $\mathcal{H}^{[i]}$, for $i \in [t]$, and $\mathcal{G}$ w.r.t. the tags, where $\mathcal{H}^{[i]}$ contains the coordinates that only appear in $\mathbf{z}^{[i]}$; $\mathcal{G}$ contains the indexes that either appear in the tags for more than once, or never appear in the tags. Later we will prove that w.h.p. all the sets $\mathcal{H}^{[i]}$, $i \in [t]$, are non-empty.
   Formally, first initialize all the sets as empty sets. Then for $(j, d) \in [\lambda] \times \mathcal{T}$:
   – If there exists an $i^* \in [t]$ such that $\mathbf{z}_j^{[i^*]} = d$ and $\forall i \neq i^*$, $\mathbf{z}_j^{[i^*]} \neq d$, then add (j, d) in $\mathcal{H}^{[i^*]}$.
   – Else, add (j, d) in $\mathcal{G}$.
2. Given the i^{th} ciphertext query, $\mathsf{MFE.Sim}(1^\lambda)$ calls for $\mathsf{F.Sim.ck}^{[i]} \leftarrow \mathsf{F.Sim}(1^{|C|})$, outputs $\mathbf{z}^{[i]}, \mathsf{F.Sim.ck}^{[i]}$ as the simulated ciphertext $\mathsf{ct}^{[i]}$.
3. Given a decryption key query on the input x with the indicators $\left\{f^{[i]}(x)\right\}_{i\in[t]}$, the mixed-FE simulator $\mathsf{MFE.Sim}(1^\lambda, x, \left\{f^{[i]}(x)\right\}_{i\in[t]})$ outputs x and $\{y_{x,j,d}\}_{j\in[\lambda],d\in\mathcal{T}}$, where each $y_{x,j,d}$ is computed in the following way:
   – First go over all $(j, d) \in \mathcal{G}$, and let $y_{x,j,d} \xleftarrow{\$} R_\lambda$, where R_λ is the range of the KHPCPRF.

– Then, for each $i \in [t]$, let $p^{[i]} := |\mathcal{H}^{[i]}|$.
For the first $p-1$ indexes $(j,d) \in \mathcal{H}^{[i]}$, let $y_{x,j,d} \xleftarrow{\$} R_\lambda$.
For the last index $(j^*, d^*) \in \mathcal{H}^{[i]}$, let

$$y_{x,j^*,d^*} := \begin{cases} \mathsf{F.Constrain.Eval}(\mathsf{F.Sim.ck}^{[i]}, x) - \sum_{j \in [\lambda], j \neq j^*} y_{x,j,z_j^{[i]}} & \text{if } f^{[i]}(x) = 0 \\ U(R_\lambda) & \text{if } f^{[i]}(x) = 1 \end{cases}. \tag{1}$$

We first prove that with all but negligible probability, all the sets $\mathcal{H}^{[i]}$, $i \in [t]$, are non-empty.

Lemma 4.6. *With probability greater or equal to* $1 - t \cdot 2^{-\lambda}$, $|\mathcal{H}^{[i]}| \geq 1$ *for all* $i \in [t]$.

Proof. For each tag $\mathbf{z}^{[i]}$, $i \in [t]$, the probability that the j^{th} coordinate of $\mathbf{z}^{[i]}$ does not show up in the other $t-1$ tags is $\geq \frac{(2t-2)-(t-1)}{2t-2} = \frac{1}{2}$. Therefore the probability that $|\mathcal{H}^{[i]}| \geq 1$ is $1-2^{-\lambda}$. Therefore with probability greater or equal to $1 - t \cdot 2^{-\lambda}$, $|\mathcal{H}^{[i]}| \geq 1$ for all $i \in [t]$.

Next we reduce the simulation security of the KHPCPRF (with the same public parameter and many independent secret keys) to the indistinguishability of the real experiment and the simulated one for the mixed-FE scheme. Suppose there is a p.p.t. adversary A that breaks the t-CT secure mixed-FE MFE with non-negligible probability η, we build a p.p.t. adversary A' for the KHPCPRF F. A' goes through the following stages.

1. Preprocessing: A' sample a set of tags $\left\{\mathbf{z}^{[i]} \xleftarrow{\$} \mathcal{T}^\lambda\right\}_{i \in [t]}$. Define the sets $\mathcal{H}^{[i]}$, for $i \in [t]$, and $\mathcal{G}$ w.r.t. the tags in the same way as was defined for the MFE simulator.
2. The mixed-FE ciphertext queries: Once the mixed-FE adversary A makes the encryption queries for $\left\{f^{[i]}\right\}_{i \in [t]}$, A' then forwards the t functions as the KHPCPRF constrained key queries. A' gets back t constrained keys $\left\{\mathsf{ck}^{[i]}\right\}_{i \in [t]}$, each of them is either a real constrained key $\mathsf{ck}^{[i]} \leftarrow \mathsf{F.Constrain}(\mathsf{F.pp}, \mathsf{F.sk}^{[i]}, f^{[i]})$ derived from some secret key $\mathsf{F.sk}^{[i]}$, or a simulated constrained key $\mathsf{ck}^{[i]} \leftarrow \mathsf{F.Sim}(1^{|C|})$.
A' then responses $\mathbf{z}^{[i]}, \mathsf{ck}^{[i]}$ to A as the ciphertext $\mathsf{ct}^{[i]}$, for $i \in [t]$.
3. The mixed-FE decryption key queries: Once the mixed-FE adversary A makes a functional decryption key query on x, A' forwards x with the indicators $\left\{f^{[i]}(x)\right\}_{i \in [t]}$ as a KHPCPRF evaluation query. A' gets back t evaluations $\left\{y^{[i]}\right\}_{i \in [t]}$, each of them is either the real evaluation $y^{[i]} = \mathsf{F.Eval}(\mathsf{F.sk}^{[i]}, x)$ on some secret key $\mathsf{F.sk}^{[i]}$, or a simulated evaluation $y^{[i]} = \mathsf{F.Sim}(x, f^{[i]}(x))$.
A' then produces the set $\{y_{x,j,d}\}_{j \in [\lambda], d \in \mathcal{T}}$ as follows:
   – First go over all $(j,d) \in \mathcal{G}$, and let $y_{x,j,d} \xleftarrow{\$} R_\lambda$, where R_λ is the range of the KHPCPRF.

– Then, for each $i \in [t]$, let $p^{[i]} := |\mathcal{H}^{[i]}|$.
  For the first $p-1$ indexes $(j,d) \in \mathcal{H}^{[i]}$, let $y_{x,j,d} \xleftarrow{\$} R_\lambda$.
  For the last index $(j^*,d^*) \in \mathcal{H}^{[i]}$, let $y_{x,j^*,d^*} = y^{[i]} - \sum_{j\in[\lambda],j\neq j^*} y_{x,j,z_j^{[i]}}$.
  A' then responses $x, \{y_{x,j,d}\}_{j\in[\lambda],d\in\mathcal{T}}$ to A as the functional decryption key for x.

4. Finally A' forwards the answer of A on whether the scheme is real or simulated.

We justify that the distributions produced by A' are computationally close to the desired distributions in the mixed-FE security game. Recall that all the sets $\mathcal{H}^{[i]}$, for $i \in [t]$, are non-empty with probability $\geq 1 - t \cdot 2^{-\lambda}$ due to Lemma 4.6.
If the KHPCPRF samples A' received are from the real distribution, then

– The correct distribution of a mixed-FE ciphertext is

$$U(\mathcal{T}^\lambda), \mathsf{F.Constrain}(\mathsf{F.pp}, \mathsf{F.sk}_f, f), \text{ where } \mathsf{F.sk}_f = \sum_{j\in[\lambda]} \mathsf{F.sk}_{j,z_j}.$$

  The distribution of the mixed-FE ciphertext produced by A' is

$$U(\mathcal{T}^\lambda), \mathsf{F.Constrain}(\mathsf{F.pp}, \mathsf{F.sk}, f), \text{ with some correctly generated secret key } \mathsf{F.sk}.$$

  These two distributions are the same due to the distribution requirement for the correctly generated secret keys for F. Recall that for $\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}'$ sampled from $\mathsf{F.skGen}(1^\lambda, \mathsf{pp})$ with independent randomness, $\mathsf{sk}_1 + \mathsf{sk}_2$ and sk' are required to be from the same distribution. This immediately implies that the sum of many correctly generated secret keys distributes the same as a single secret key.
– The correct mixed-FE functional decryption key for x is $\mathsf{sk}_x = x, \{\mathsf{F.Eval}(\mathsf{F.sk}_{j,d}, x)\}_{j\in[\lambda],d\in\mathcal{T}}$. We argue that the mixed-FE functional decryption key for x produced by A' is computationally indistinguishable to the real one due to the pseudorandomness of the PRF evaluations w.r.t. the secret keys whose constrained keys are not giving out.
  - For all $(j,d) \in \mathcal{G}$, the PRF secret keys on these indexes are independent from the constrained keys that are given out, so the PRF evaluations on these indexes are indistinguishable from random.
  - For each $i \in [t]$, pick an index $(j^*,d^*) \in \mathcal{H}^{[i]}$, the real PRF evaluation y_{x,j^*,d^*} can be re-written following the key-homomorphism as

$$\mathsf{F.Eval}(\mathsf{F.sk}_f, x) - \sum_{j\in[\lambda],j\neq j^*} \mathsf{F.Eval}(\mathsf{F.sk}_{j,z_j^{[i]}}, x) = \mathsf{F.Eval}((\mathsf{F.sk}_f - \sum_{j\in[\lambda],j\neq j^*} \mathsf{F.sk}_{j,z_j^{[i]}}), x).$$

    Therefore, y_{x,j^*,d^*} distributes correctly due to the distribution requirement of the KHPCPRF secret keys. The PRF evaluations on the rest of the indexes in $\mathcal{H}^{[i]}$ are using independent PRF secret keys, so these evaluations are pseudorandom.

If the KHPCPRF samples A' received are from the simulated distribution, then

- The correct simulated distribution of the mixed-FE ciphertexts is $U(\mathcal{T}^\lambda), \mathsf{F.Sim}(1^{|C|})$, which is exactly what A' produces.
- For the simulated mixed-FE functional decryption key for x. Observe that the constrained PRF simulator outputs $U(R_\lambda)$ if $f(x) = 1$, outputs $\mathsf{F.Constrain.Eval}(\mathsf{F.Sim.ck}, x)$ if $f(x) = 0$. So the functional decryption key produced by A' follows the correct distribution.

Hence A' wins with $\eta - \mathrm{negl}(\lambda)$ advantage in the KHPCPRF simulation security game.

Finally we verify the public/secret-key mode indistinguishability. It follows from observing that if $f^{[i]}(x) = 1$ for all x being queried, the simulated ciphertext is independent from the simulated functional decryption keys, and has the same distribution as the public-key mode.

The Instantiation from the LWE-Based Almost-Key-Homomorphic PCPRFs. We provide the details for instantiating the mixed-FE from the LWE-based 1-almost-key-homomorphic PCPRFs. Note that the maximum number of key addition is λ in both the construction and the analysis, so we can choose the modulus p to be $\geq 4\lambda$, the range R as $\mathbb{Z}_p^n$ where $n = \Omega(\lambda)$, and the rest of the parameters under the restrictions mentioned in the original PCPRF constructions.

In the construction of the mixed-FE, we change the decryption algorithm as: $\mathsf{MFE.Dec}(\mathsf{sk}_x, \mathsf{ct}_f)$ parses sk_x as $x, \{y_{j,d}\}_{j\in[\lambda],d\in\mathcal{T}}$ and ct_f as $\mathbf{z}, \mathsf{ck}_f$, outputs

$$\begin{cases} 0 & \text{if } \left\| \mathsf{F.Constrain.Eval}(\mathsf{ck}_f, x) - \left(\sum_{j\in[\lambda]} y_{j,z_j}\right) \right\|_\infty \leq \lambda \\ 1 & \text{else} \end{cases}.$$

In the simulation, we change one piece in the simulated functional decryption key for each $i \in [t]$. That is, for the last index $(j^*, d^*) \in \mathcal{H}^{[i]}$, we let

$$y_{x,j^*,d^*} := \begin{cases} \mathsf{F.Constrain.Eval}(\mathsf{F.Sim.ck}^{[i]}, x) + N(\lambda) - \sum_{j\in[\lambda], j\neq j^*} y_{x,j,z_j^{[i]}} & \text{if } f^{[i]}(x) = 0 \\ U(R_\lambda) & \text{if } f^{[i]}(x) = 1 \end{cases}. \quad (2)$$

where $N(\lambda)$ is a noise factor added to compensate the error caused by the almost-hey-homomorphism. The distribution of $N(\lambda)$ is efficiently sampleable and identical to the distribution of

$$\sum_{j\in[\lambda]} \mathsf{F.Eval}(\mathsf{F.sk}_j, x) - \mathsf{F.Eval}(\mathsf{F.sk}_\Sigma, x)$$

where $\left\{\mathsf{F.sk}_j \leftarrow \mathsf{F.skGen}(1^\lambda, \mathsf{F.pp})\right\}_{j\in[\lambda]}$ and $\mathsf{F.sk}_\Sigma = \sum_{j\in[\lambda]} \mathsf{F.sk}_j$.

5 Attribute-Based Traitor Tracing

5.1 Definition of Attribute-Based Traitor Tracing

Definition 5.1 (Attribute-Based Traitor Tracing (AB-TT)). *An* attribute-based traitor-tracing *(AB-TT) scheme for a class of functions* $\mathcal{F}_\mu = \{f : \{0,1\}^\mu \to \{0,1\}\}$ *and a message length* ℓ *(where* μ, ℓ *are functions of the security parameter* λ*) is a tuple of p.p.t. algorithms* $(\mathsf{Setup}, \mathsf{skGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *such that:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter* 1^λ*, outputs the master secret key* msk *and the public parameter* pp*.*
- $\mathsf{skGen}(\mathsf{msk}, f, i)$ *takes* msk*, a function* $f \in \mathcal{F}_\mu$ *and an identity* $i \in [2^\lambda]$ *and outputs a decryption key* $\mathsf{sk}_{f,i}$*.*
- $\mathsf{Enc}(\mathsf{pp}, x, m)$ *takes as input* pp*, the attribute* $x \in \{0,1\}^\mu$ *and a message* $m \in \{0,1\}^\ell$ *and outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{sk}_{f,i}, \mathsf{ct})$ *takes* $\mathsf{sk}_{f,i}$ *and* ct*, outputs the message* m *or* $\perp$*.*
- $\mathsf{Trace}^D(\mathsf{msk}, 1^n, 1^h, x, m_0, m_1)$ *takes as input* msk *the number of identities* $n \in \mathbb{Z}$*, a correctness parameter* h*, an attribute* $x \in \{0,1\}^\mu$ *and two messages* m_0, m_1 *as well as oracle-access to a "decoder"* D*. Outputs an identity* $t \in [2^\lambda]$ *or* $\perp$ *to indicate that no identity was traced.*

The scheme satisfies the following properties:

- *Correctness: This is the same as in standard ABE if we ignore the index* i *(allow it to be arbitrary).*
- *ABE Security: This is the same as in standard ABE if we ignore the index* i *(allow the adversary to choose it arbitrarily).*
- *Tracing Security: For any* $\varepsilon(\lambda) = 1/\operatorname{poly}(\lambda)$*. We define the following experiment between an adversary* A *and a challenger:*
  1. $1^n \leftarrow \mathsf{A}(1^\lambda)$*.*
  2. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
  3. $(D, x, m_0, m_1) \leftarrow \mathsf{A}^{\mathsf{skGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{pp})$ *:* $x \in \{0,1\}^\mu, m \in \{0,1\}^\ell$ *and oracle queries* (f, t) *must satisfy* $f \in \mathcal{F}_\mu$ *and* $t \in [n]$*.*
  4. $t \leftarrow \mathsf{Trace}^D(\mathsf{msk}, 1^n, 1^{\lceil 1/\varepsilon \rceil}, x, m_0, m_1)$

  Within the above experiment, define the event $\mathsf{GoodDecoder}$ *to occur if*

$$\Pr[D(\mathsf{ct}) = b \ : \ b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m_b)] \geq 1/2 + \varepsilon(\lambda)$$

  where D, x, m_0, m_1 *are defined in step 3 of the experiment. Define the event* $\mathsf{BadTrace}$ *to occur if, for the* t *output by the trace algorithm in step 4, the adversary never made a* skGen *query of the form* (f, t) *where* $f(x) = 1$*. We require that* $\Pr[\mathsf{GoodDecoder} \wedge \mathsf{BadTrace}] \leq \operatorname{negl}(\lambda)$*.*

We make several remarks about the above definition. Firstly, while syntactically the scheme allows the identities i to come from a large space $[2^\lambda]$, for tracing security we assume that the range of identities $[n]$ is polynomially sized where the polynomial can be chosen arbitrarily by the adversary. Secondly, we think of the identities i as corresponding to users but each user can get several different keys $\mathsf{sk}_{f,i}$ for different functions f.

5.2 Tool: Attribute-Based Mixed FE

An attribute-based Mixed FE (AB-MFE) combines aspects of MFE and ABE. In particular, like in ABE, a secret key is associated with an ABE function f and a (public-key) ciphertext is associated with an ABE attribute x and a message m and decryption works if $f(x) = 1$. However, like in MFE, the secret key is also associated with an MFE function g. The MFE function is irrelevant when decrypting public-key ciphertexts. But there is also a secret-key encryption algorithm that additionally associates a ciphertext with an MFE attribute y. A secret-key ciphertext decrypts correctly if $f(x) = 1$ and $g(y) = 1$. The security requirements are a combination of MFE and ABE security.

(Note that, from the point of view of MFE, we switched the role of attributes and functions from the original definition by associating secret keys with functions and ciphertexts with attributes. This change is essentially cosmetic to better fit the connection with ABE and one can convert back and forth easily using universal circuits).

Definition 5.2 (Attribute-Based Mixed FE). *An* attribute-based mixed-FE *(AB-MFE) scheme for a class of ABE functions $\mathcal{F}_\mu = \{f : \{0,1\}^\mu \to \{0,1\}\}$, MFE functions $\mathcal{G}_\nu = \{g : \{0,1\}^\nu \to \{0,1\}\}$ and message length ℓ (where μ, ν, ℓ are functions of the security parameter λ) is a tuple of p.p.t. algorithms* $(\mathsf{Setup}, \mathsf{skGen}, \mathsf{pkEnc}, \mathsf{skEnc}, \mathsf{Dec})$ *such that:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter 1^λ, outputs the master secret key* msk *and the public parameter* pp.
- $\mathsf{skGen}(\mathsf{msk}, f, g)$ *takes* msk*, an ABE function $f \in \mathcal{F}_\mu$, a MFE function $g \in \mathcal{G}_\nu$ and outputs a decryption key* $\mathsf{sk}_{f,g}$.
- $\mathsf{pkEnc}(\mathsf{pp}, x, m)$ *takes as input* pp*, the ABE attribute $x \in \{0,1\}^\mu$ and a message $m \in \{0,1\}^\ell$ and outputs a ciphertext* ct.
- $\mathsf{skEnc}(\mathsf{msk}, x, y, m)$ *takes as input* pp*, the ABE attribute $x \in \{0,1\}^\mu$ the MFE attribute $y \in \{0,1\}^\nu$ and a message $m \in \{0,1\}^\ell$ and outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk}_{f,g}, \mathsf{ct})$ *takes* $\mathsf{sk}_{f,g}$ *and* ct*, outputs a message m or $\perp$.*

The scheme is q-query secure for some polynomial $q = q(\lambda)$ if it satisfies the following properties:

- ***Correctness:*** *For all $f \in \mathcal{F}_\mu$ and all $x \in \{0,1\}^\mu$ such that $f(x) = 1$, for all $g \in \mathcal{G}_\nu$ and all $m \in \{0,1\}^\ell$ it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{f,g}, \mathsf{ct}) = m \;:\; \begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda),\\ \mathsf{sk}_{f,g} \leftarrow \mathsf{skGen}(\mathsf{msk}, f, g),\\ \mathsf{ct} \leftarrow \mathsf{pkEnc}(\mathsf{pp}, x, m)\end{array}\right] \geq 1 - \mathrm{negl}(\lambda)$$

  Furthermore, for all f, x, m as above and all $y \in \{0,1\}^\nu$ such that $g(y) = 1$ it holds that:

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{f,g}, \mathsf{ct}) = m \;:\; \begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda),\\ \mathsf{sk}_{f,g} \leftarrow \mathsf{skGen}(\mathsf{msk}, f, g),\\ \mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{pp}, x, y, m)\end{array}\right] \geq 1 - \mathrm{negl}(\lambda)$$

- ***ABE Security:*** *The algorithms* $(\mathsf{Setup}, \mathsf{skGen}, \mathsf{pkEnc}, \mathsf{Dec})$ *satisfy ABE security if we ignore the* g *part of* skGen.
- ***Public/Secret Hiding:*** *Consider the following experiment with an adversary* A
  1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
  2. $(x^*, y^*, m^*) \leftarrow \mathsf{A}^{\mathsf{skGen}(\mathsf{msk},\cdot,\cdot),\mathsf{skEnc}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{mpk})$
  3. $b \leftarrow \{0,1\}$. *If* $b = 0$ *then set* $\mathsf{ct} \leftarrow \mathsf{pkEnc}(\mathsf{pp}, x^*, m^*)$ *else if* $b = 1$ *set* $\mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{msk}, x^*, y^*, m^*)$.
  4. $b' \leftarrow \mathsf{A}(\mathsf{ct})$.

  An adversary A *in the above experiment is* legal *if (a) it makes at most* q *queries to the* skEnc *oracle, and (b) every query* (f, g) *made to the* skGen *oracle satisfies* $g(y^*) = 1$*, meaning that the MFE component is always qualified to decrypt. We require that for any legal* A *in the above game we have* $\Pr[b' = b] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$.
- ***MFE Attribute Hiding:*** *Consider the following experiment with an adversary* A
  1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
  2. $(x^*, m^*, y_0, y_1) \leftarrow \mathsf{A}^{\mathsf{skGen}(\mathsf{msk},\cdot,\cdot),\mathsf{skEnc}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{mpk})$
  3. $b \leftarrow \{0,1\}$, $\mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{msk}, x^*, y_b, m^*)$.
  4. $b' \leftarrow \mathsf{A}(\mathsf{ct})$.

  An adversary A *in the above experiment is* legal *if (a) it makes at most* q *queries to the* skEnc *oracle, and (b) every query* (f, g) *made to the* skGen *oracle satisfies* $g(y_0) = g(y_1)$*. We require that for any legal* A *in the above game we have* $\Pr[b' = b] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$.
- ***Message Hiding:*** *Consider the following experiment with an adversary* A
  1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
  2. $(x^*, y^*, m_0, m_1) \leftarrow \mathsf{A}^{\mathsf{skGen}(\mathsf{msk},\cdot,\cdot),\mathsf{skEnc}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{mpk})$
  3. $b \leftarrow \{0,1\}$, $\mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{msk}, x^*, y^*, m_b)$.
  4. $b' \leftarrow \mathsf{A}(\mathsf{ct})$.

  An adversary A *in the above experiment is* legal *if (a) it makes at most* q *queries to the* skEnc *oracle, and (b) every query* (f, g) *made to the* skGen *oracle satisfies* $g(y^*) = 0$ *or* $f(x^*) = 0$*. We require that for any legal* A *in the above game we have* $\Pr[b' = b] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$.

Decoder-Based Security. In the above definition of AB-MFE security we considered three security properties each of which consists of a 4-step experiment. For each of them, the adversary can make at most q queries to $\mathsf{skEnc}(\mathsf{msk}, \cdot)$ oracle during the experiment and at the end of the experiment gets as input a ciphertext ct and outputs a bit b. We now consider a variant of the three security properties which we call *decoder-based security*. Firstly, the adversary loses access to the $\mathsf{skEnc}(\mathsf{msk}, \cdot)$ oracle entirely in each of the experiments. Secondly, in step 2 of each of the experiments the adversary additionally outputs a decoder circuit D and the experiment ends. For some $\varepsilon = \varepsilon(\lambda)$, we say that the decoder is ε-good if $\Pr[D(\mathsf{ct}) = b] \geq 1/2 + \varepsilon$ where b and ct are sampled as in step 3 of each of the original experiments. For decoder-based security we will require that in

each of the experiments, for any legal adversary A and for any $\varepsilon(\lambda) = 1/\operatorname{poly}(\lambda)$ it holds that

$$\Pr[D \text{ is } \varepsilon(\lambda)\text{-good}] \leq \operatorname{negl}(\lambda)$$

where D is the output of the adversary in step 2 of the experiment.

The reason for defining both standard and decoder-based security properties is that the standard definitions are more natural to target when constructing Attribute-Based Mixed-FE, while the decoder definitions are directly compatible with tracing definitions. The lemma below connects them, allowing us to get the best of both worlds.

Lemma 5.3. *An AB-MFE with $(q = 1)$-query security also satisfies decoder-based security.*

A variant of the above lemma for MFE was given in [20] (Sect. 4). The proof of our lemma for AB-MFE is identical, up to minor syntactic changes needed to account for the expanded ABE syntax.

5.3 From Attribute-Based Mixed-FE to Attribute-Based Traitor Tracing

We now move on to building Attribute-Based Traitor Tracing from Attribute-Based Mixed-FE using the decoder-based security properties. We begin with some high level intuition. Suppose an attacker produces a decoder D that can decrypt ciphertexts associated with some attribute x. A natural approach would be to follow [20] using the Mixed-FE piece to remove each user one index at a time until we reach an index i where the decryption probability between encryptions to index i and $i+1$ differ. At this point we can finger user i as having contributed to creating the box D. However, the problem with this strategy is that the decoder algorithm might catch (and only catch) a user i who was not qualified to decrypt the ABE ciphertext to begin with. As argued earlier a meaningful trace will catch a user with a private key for f where $f(x) = 1$.

For that reason the MixedFE component will be used to gradually remove *only* qualified decryptors one index at a time. That is the function g will be of the form if $f(x) = 0$ or $j \geq i$. So a user with index i and $f(x) = 0$ will always have the Mixed-FE component output 1 even if $j \geq i$. Therefore if there is some index i where the decoding probability differs between encryptions to index $i - 1$ and i it must be the case that user i was a contributor and was qualified to decrypt. This is perhaps slightly counterintuitive as our tracing strategy explicitly always allows non-qualified users to pass the Mixed-FE portion.

We observe that for any good decoder box there must be some such i. The public/secret hiding property guarantees that any good decryptor box for the public key encryption will still decrypt well on index $i = 0$. The Message hiding property guarantees that when encrypting to index $i = n$ that no user will be able to decrypt. This is either because $f(x) = 0$ or due to the way g was selected. Thus there must exists some i where the decoder has a non-negligible gap in decrypting.

A formal description of the tracing system appears below.

Let $\mathcal{F}_\mu = \{f : \{0,1\}^\mu \to \{0,1\}\}$ be a function family. Define the family $\mathcal{G}_\nu = \{g : \{0,1\}^\nu \to \{0,1\}\}$ consisting of functions

$$g_{f,i}(x,j) = \begin{cases} 1 & \text{if } f(x) = 0 \text{ or } j \geq i \\ 0 & \text{otherwise} \end{cases}$$

where $i \in [2^\lambda]$, $j \in [2^\lambda] \cup \{0\}$ and $f \in \mathcal{F}_\mu$.

Assume that $\mathsf{ABMFE} = (\mathsf{Setup}, \mathsf{skGen}, \mathsf{pkEnc}, \mathsf{skEnc}, \mathsf{Dec})$ is an AB-MFE for the class of ABE functions $\mathcal{F}_\mu$ and the class of MFE functions $\mathcal{G}_\nu$. Further assume that ABMFE satisfies decoder-based security.

We show how to construct an AB-TT scheme $\mathsf{ABTT} = (\mathsf{Setup}', \mathsf{skGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Trace})$ for the function class $\mathcal{F}_\mu = \{f : \{0,1\}^\mu \to \{0,1\}\}$ as follows.

- Setup' is the same as Setup.
- $\mathsf{skGen}'(\mathsf{msk}, f, i)$: Construct $g_{f,i} \in \mathcal{G}_\nu$ and let $\mathsf{sk}_{f,i} \leftarrow \mathsf{skGen}(\mathsf{msk}, f, g_{f,i})$.
- Enc' is the same as pkEnc.
- Dec' is the same as Dec.
- $\mathsf{Trace}^D(\mathsf{msk}, 1^n, 1^h, x, m_0, m_1)$: Let $\varepsilon = 1/h$ and $W = \lambda \cdot (n \cdot h)^2$. For $i = 0$ to n, the trace algorithm does the following:
  1. It first sets $c_i := 0$. For $j = 1$ to W, it does the following:
     (a) It chooses $b_{i,j} \leftarrow \{0,1\}$, sets $\mathsf{ct}_{i,j} \leftarrow \mathsf{skEnc}(\mathsf{msk}, x, (x,i), m_{b_{i,j}})$. If $D(\mathsf{ct}_{i,j}) = b_{i,j}$, it sets $c_i = c_i + 1$.
  2. It sets $\hat{p}_i = c_i/W$.

  The trace algorithm outputs the first index $i \in \{1, 2, \ldots, n\}$ such that $\hat{p}_{i-1} - \hat{p}_i \geq \varepsilon/4n$. If no such index exists output $\perp$.

Theorem 5.4. *For any $\mathcal{F}_\mu$ with a corresponding $\mathcal{G}_\nu$, if* ABMFE *is a secure AB-MFE for the ABE class $\mathcal{F}_\mu$ and the MFE class $\mathcal{G}_\nu$ satisfying decoder-based security then the scheme* ABTT *is a secure attribute-based traitor tracing scheme (AB-TT).*

A variant of the above theorem showing that (non attribute-based) MFE implies traitor-tracing was given in [20] (Sect. 4.2.2). The proof of our theorem for AB-MFE is essentially identical. We give a high level proof below.

Proof. Assume that, in the tracing game, the adversary outputs a good decoder D meaning that the event $\mathsf{GoodDecoder}$ occurs. This means D can find b given $\mathsf{pkEnc}(\mathsf{pp}, x, m_b)$ with some noticeable advantage ε.

- By (decoder-based) public/secret hiding, it must be the case that D can find b given $\mathsf{skEnc}(\mathsf{msk}, x, (x,0), m_b)$ with advantage $\varepsilon - \mathrm{negl}(\lambda)$. Otherwise D could distinguish between $\mathsf{pkEnc}(\mathsf{pp}, x, m_b)$ and $\mathsf{skEnc}(\mathsf{msk}, x, (x,0), m_b)$ even though for all f, i we have $g_{f,i}(x,0) = 1$.
- By (decoder-based) message hiding, D can only have negligible advantage in finding b given $\mathsf{skEnc}(\mathsf{msk}, x, (x, n+1), m_b)$. Not that for every AB-MFE secret key obtained by the adversary associated with functions $(f, g_{f,i})$ we have that either $f(x) = 0$ or $g_{f,i}(x, n+1) = \neg f(x) = 0$.

- Combining the above two points, there must be at least one index j such that the advantage of D in finding b given $\mathsf{skEnc}(\mathsf{msk}, x, (x, j), m_b)$ is at least $(\varepsilon - \mathrm{negl}(\lambda))/n \geq \varepsilon/(2n)$ larger for j versus $j+1$. We can use the Chernoff bound to argue that, with overwhelming probability, the tracing algorithm outputs some t for which the difference in advantage is at least $\varepsilon'/8n$. For this t there must be at least one $b \in \{0,1\}$ such that the decoder D can distinguish between $\mathsf{skEnc}(\mathsf{msk}, x, (x, t), m_b)$ and $\mathsf{skEnc}(\mathsf{msk}, x, (x, t+1), m_b)$ with noticeable advantage.
- By (decoder-based) attribute-hiding security, the above can only happen if the adversary got an AB-MFE secret key for the functions $(f, g_{f,j})$ such that $g_{f,j}(x,t) \neq g_{f,j}(x,t+1)$, which can only happen if $f(x) = 1$ and $j = t$. This means that the adversary must have queried an AB-TT secret key for the function f such that $f(x) = 1$ with the identity t. Therefore the tracing algorithm succeeds in finding a valid traitor t and the event $\mathsf{BadTrace}$ does not occur whenever $\mathsf{GoodDecoder}$ occurs as we wanted to show.

5.4 From Mixed-FE to Attribute-Based Mixed-FE

Let $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.skGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ be an ABE scheme for all circuits. Let $\mathsf{MFE} = (\mathsf{MFE.Setup}, \mathsf{MFE.skGen}, \mathsf{MFE.pkEnc}, \mathsf{MFE.skEnc}, \mathsf{MFE.Dec})$ be an MFE scheme (without a message) for some function class $\mathcal{F}_\mu = \{f : \{0,1\}^\mu \to \{0,1\}\}$; for simplicity we switch the roles of attribute/function and associate keys with functions and ciphertexts with attributes. We construct and AB-MFE scheme $\mathsf{ABMFE} = (\mathsf{Setup}, \mathsf{skGen}, \mathsf{pkEnc}, \mathsf{skEnc}, \mathsf{Dec})$ as follows:

- Setup: Run $(\mathsf{ABE.pp}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and $(\mathsf{MFE.pp}, \mathsf{MFE.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$. Output $\mathsf{pp} = (\mathsf{ABE.pp}, \mathsf{MFE.pp})$ and $\mathsf{msk} = (\mathsf{ABE.msk}, \mathsf{MFE.msk})$.
- $\mathsf{skGen}(\mathsf{msk}, f, g)$: Let $\mathsf{MFE.sk}_g \leftarrow \mathsf{MFE.skGen}(\mathsf{MFE.msk}, g)$. Let C be a circuit which has $f, \mathsf{MFE.sk}_g$ hard-coded inside it, takes as input $x, \mathsf{MFE.ct}$ and outputs 1 if $f(x) = 1$ and $\mathsf{MFE.Dec}(\mathsf{MFE.sk}_g, \mathsf{ct}) = 1$. Output $\mathsf{sk}_{f,g} \leftarrow \mathsf{ABE.skGen}(\mathsf{ABE.msk}, C)$.
- $\mathsf{pkEnc}(\mathsf{pp}, x, m)$: Let $\mathsf{MFE.ct} \leftarrow \mathsf{MFE.pkEnc}(\mathsf{MFE.pp})$. Output $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pp}, (x, \mathsf{MFE.ct}), m)$.
- $\mathsf{skEnc}(\mathsf{msk}, x, y, m)$: Let $\mathsf{MFE.ct} \leftarrow \mathsf{MFE.skEnc}(\mathsf{MFE.msk}, y)$. Output $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pp}, (x, \mathsf{MFE.ct}), m)$.
- $\mathsf{Dec}(\mathsf{sk}_{f,g}, \mathsf{ct})$: Output $\mathsf{ABE.Dec}(\mathsf{sk}_{f,g}, \mathsf{ct})$.

Theorem 5.5. *If* ABE *is a secure ABE scheme and* MFE *is a secure MFE scheme then* ABMFE *is a secure AB-MFE scheme.*

Proof. The correctness of the AB-MFE follows directly from that of the ABE and MFE schemes. The ABE security of the AB-MFE follows directly from the ABE security of the ABE. The "public/secret hiding" security of the AB-MFE follows directly from that of the MFE. The "attribute-hiding" security of the AB-MFE follows directly from the "attribute-hiding" security of the MFE (previously we called this "function hiding" but since we switched the roles of

attributes and functions this is now "attribute hiding"). Lastly for message-hiding security of the AB-MFE we rely on the security of the ABE. In particular, the adversary gets as a challenge an ABE ciphertext with attribute $(x^*, \mathsf{MFE.ct} = \mathsf{MFE.skEnc}(\mathsf{MFE.msk}, y^*))$ but only has ABE keys for circuits C such that $C(x, \mathsf{MFE.ct}) = 1$ if $f(x^*) = 1$ and $\mathsf{MFE.Dec}(\mathsf{MFE.sk}_g, \mathsf{ct}) = 1 \Leftrightarrow g(y^*) = 1$. Therefore if one of $f(x^*) = 0$ or $g(y^*) = 0$ always holds, it must mean that none of the ABE secret keys are qualified to decrypt the challenge ciphertext.

Acknowledgments. The research of Yilei Chen was conducted at Boston University supported by the NSF MACS project and NSF grant CNS-1422965. Vinod Vaikuntanathan is supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, the NEC Corporation and a Steven and Renee Finn Career Development Chair from MIT. This work was also sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236. Brent Waters is supported by NSF CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Hoeteck Wee is supported by ERC Project aSCEND (H2020 639554). Daniel Wichs is supported by NSF grants CNS-1314722, CNS-1413964.

References

1. Abdalla, M., Dent, A.W., Malone-Lee, J., Neven, G., Phan, D.H., Smart, N.P.: Identity-based traitor tracing. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 361–376. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_24
2. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 3–35. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_1
3. Agrawal, S., Bhattacherjee, S., Phan, D.H., Stehlé, D., Yamada, S.: Efficient public trace and revoke from standard assumptions: extended abstract. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30–November 03 2017, pp. 2277–2293 (2017)
4. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 173–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_7
5. Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_23
6. Boneh, D., Lewi, K., Wu, D.J.: Constraining pseudorandom functions privately. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 494–524. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_17
7. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_34

8. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, I October 30–November 3 2006, pp. 211–220 (2006)
9. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_27
10. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 575–584. ACM (2013)
11. Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and More) from LWE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 264–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_10
12. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 446–476. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_16
13. Chen, Y., Vaikuntanathan, V., Wee, H.: GGH15 beyond permutation branching programs: proofs, attacks, and candidates. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 577–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_20
14. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_25
15. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_3
16. Garg, S., Kumarasubramanian, A., Sahai, A., Waters, B.: Building efficient fully collusion-resilient traitor tracing and revocation schemes. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 121–130 (2010)
17. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 498–527. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_20
18. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11
19. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: FOCS, pp. 612–621 (2017)
20. Goyal, R., Koppula, V., Waters, B.: Collusion resistant traitor tracing from learning with errors. In: STOC (2018)
21. Katz, J., Schröder, D.: Bootstrapping obfuscators via fast pseudorandom functions. In: Annual Conference of the ITA (ACITA) (2011)
22. Liu, Z., Cao, Z., Wong, D.S.: Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on eBay. In: 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, Berlin, Germany, 4–8 November 2013, pp. 475–486 (2013)

23. Liu, Z., Wong, D.S.: Practical ciphertext-policy attribute-based encryption: traitor tracing, revocation, and large universe. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 127–146. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_7
24. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: STOC, pp. 333–342 (2009)
25. Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of Ring-LWE for any ring and modulus. In: STOC, pp. 461–473. ACM (2017)
26. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 1–40 (2009)
27. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, 4–8 October 2010, pp. 463–472. ACM (2010)
28. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: FOCS, pp. 600–611 (2017)