Catching Many Traitors in Threshold Traitor Tracing: Lower Bounds and Constructions

Dan Boneh, Aditi Partap, and Mark Zhandry

Stanford University
{dabo,aditi712,mzhandry}@cs.stanford.edu

Abstract. A t-out-of-n threshold decryption scheme distributes decryption key shares among n parties so that any t of them can jointly decrypt a ciphertext, while fewer than t learn nothing about the plaintext. Traditional threshold schemes provide no accountability: a coalition of t or more parties can combine their key shares and construct a *pirate decoder* that decrypts arbitrary well-formed ciphertexts, without any risk of being traced. To address this, Boneh, Partap, and Rotem [CRYPTO '24] introduced the notion of *threshold traitor tracing* (TTT), where a tracing algorithm that is given black-box access to the pirate decoder can identify at least one of the colluding parties. Many subsequent threshold traitor tracing schemes similarly find only a single traitor, even though the decoder must have been constructed using at least t keys. While some constructions can find multiple traitors, they do so at the cost of large ciphertexts or only achieving a weak form of correctness.

In this work, we make the following contributions:

- *Lower bounds*: We show that for all existing traitor tracing techniques, the ciphertext must be large to allow tracing close to t traitors. In particular, to trace $t - O(1)$ traitors, the ciphertext size must be at least $\Omega(t)$. To trace $a \leq t - \omega(1)$ traitors, the ciphertext size must scale with $\Omega(\frac{a-1}{t-a+1})$. For schemes that rely on fingerprinting codes, we show an even stronger lower bound.
- *Upper bounds*: We present two generic compilers that construct traitor tracing for general access structures (beyond threshold) from two building blocks: attribute based encryption for general access structures and sufficiently-expressive policies and mixed functional-encryption. We also present two concrete instantiations. Under exponential security assumptions, we construct a pairings-based threshold traitor tracing scheme that can trace t traitors with ciphertext size $O(t^2)$. We also construct an LWE-based traitor tracing scheme for a DNF access structure, that can trace an authorized subset of traitors with ciphertext size $O(\hat{t}^2)$, where $\hat{t}$ denotes the size of the largest unauthorized subset in the access structure.
- *A Candidate Theoretical Instantiation*: We present a new tracing mechanism that can trace $t(1 - 1/\lambda^c)$ traitors with $\mathsf{poly}(\lambda)$ size ciphertext, public key, and secret keys. We prove security assuming ideal (black box) obfuscation.

Our work raises several open questions in the context of tracing multiple parties in a threshold traitor tracing scheme.

1 Introduction

Traitor tracing [CFN94] is an encryption system where the key generation algorithm outputs a public key and n secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$, one key per user. Every secret key, on its own, can decrypt a well-formed ciphertext. The central property is that if a coalition of parties $\mathcal{J} \subseteq [n]$ use their secret keys $\{\mathsf{sk}_i\}_{i \in \mathcal{J}}$ to build a decryption algorithm D, then this D can be traced to at least one of the parties in $\mathcal{J}$. The tracing algorithm has black-box access to D and should output the index of a party $i \in \mathcal{J}$ with overwhelming probability. Constructions try to optimize the public key, secret key, and ciphertext sizes. A number of impressive works give constructions where all three grow at most poly-logarithmically in the number of users n [GKW18, CVW+18, Zha25, BZ14]. Some constructions use fingerprinting codes [BS95, Tar03] to achieve constant size ciphertexts (i.e., independent of n), but then the secret key size is $\Omega(n)$ [BN08, BP08].

Traitor tracing is intended to deter users from creating a decryption algorithm D that they can sell to interested parties. If they do, the tracing algorithm applied to D will catch at least one of them. Clearly the tracing algorithm cannot be required to catch more than one member of $\mathcal{J}$. Indeed, if only one secret key in $\mathcal{J}$ is sufficient to construct D, then the tracing algorithm can at most trace D to that single key. Other members of $\mathcal{J}$ cannot be reliably blamed.

Threshold traitor tracing. With the growing use of threshold decryption, Boneh, Partap, and Rotem [BPR24a] defined the notion of *Threshold Traitor Tracing* (TTT) — traitor tracing for threshold decryption. Here again the key generation algorithm outputs a public key and n secret keys. However, t secret keys are needed to decrypt a well-formed ciphertext, while $t-1$ secret keys cannot. If a coalition of t or more users get together to create a decryption algorithm D, then a tracing algorithm can trace D to at least one member of the coalition that created D. This again is meant to deter decryption parties from creating and selling a decryption algorithm to interested parties. Boneh, Partap, and Rotem [BPR24a] gave several efficient constructions for TTT. Subsequent work strengthened the TTT security model to provide chosen ciphertext security [BBF+25], public tracing [CPP25], silent setup [BHO25, CMMS25], and poly-log parameters [DDP+25].

In all these works, however, the tracing algorithm is only required to trace D to a *single* member of $\mathcal{J}$. In the context of threshold traitor tracing, a decryption algorithm D must be constructed from at least t secret keys. Therefore, it is reasonable to require the tracing algorithm to output at least t traitors. This is a much harder problem than tracing to a single party. If it can be done, it provides a stronger deterrence against selling a pirate decryption algorithm. Moreover, tracing t traitors is what one would expect from threshold traitor tracing.

In this paper, we focus on the problem of tracing t traitors, which we call multi-tracing. Some constructions already support multi-tracing, but with some limitations. The works of [BPR24a] and [BHO25] point out that the trivial threshold traitor tracing scheme, with $O(n)$ size ciphertexts, supports tracing D to at least t members of $\mathcal{J}$. Bormet, Faust, and Othman [BFO26] show how to trace $\Theta(t)$ traitors with sub-linear ciphertexts, but only achieve a relaxed notion of correctness: decryption is guaranteed for a random set of t parties and every set of $n - \sqrt{n}$ parties, but not for every set of t parties [1].

The key question is whether multi-tracing can be done with short ciphertexts and complete correctness.

1.1 Our Results: Lower Bounds

We begin with two lower bounds on the ciphertext size for threshold traitor tracing schemes that trace D to multiple users. These lower bounds correspond to two different tracing techniques: subset-based tracing and tracing based on fingerprinting codes.

Subset-based Tracing We first consider a restricted but very general class of threshold traitor tracing algorithms that we call "subset-based." In such a scheme, every ciphertext c is associated with a certain subset of parties $S_x \subseteq [n]$. A set of t parties $\mathcal{J} \subseteq [n]$ can decrypt c if and only if $\mathcal{J} \subseteq S_x$. Normally, one encrypts a message μ with respect to the entire set of parties $[n]$, so that every subset of t parties can decrypt. However, during tracing, the tracing algorithm will

[1] Their second construction allows any set of $n/2 + cn$ parties to decrypt (for a constant $c < 1/2$) with sub-linear ciphertexts, but can only trace $\Theta(\sqrt{n})$ traitors.

query the decryption algorithm D with ciphertexts targeting different subsets S_x, and then observe whether D successfully decrypts or not. This approach captures *all* existing threshold traitor tracing schemes [BPR24a, DDP+25, CMMS25, BHO25, BBF+25, BFO26], as well as existing non-threshold traitor tracing schemes (e.g., [BN08, BSW06, GLW23, Zha21, GKW18, CVW+18, Zha25, BZ14]) that do not make any stipulations about the decoder program.

Theorem 1 (Informal). *For a threshold traitor tracing scheme with subset-based tracing for n parties and threshold t,*

- *If the scheme can trace $t - O(1)$ traitors, then the ciphertext size must grow with $t \log(\lambda)$.*
- *If the scheme can trace $a \leq t - \omega(1)$ traitors, for large n, the ciphertext size must scale with $\frac{a-1}{t-a+1}\omega(\log \lambda)$.*
- *If the scheme is universal, meaning the tracing algorithm is independent of the decoder box (and in particular, traces boxes running in super-polynomial time), then, the ciphertext size must scale with $(a-1)\log_2(n/t)$.*

Fingerprinting Code-based Tracing Fingerprinting codes, originally introduced in [BS95], have been used to build traitor tracing [BN08, BP08, Zha20] and more recently, threshold traitor tracing [BPR24a, BBF+25], albeit for catching one traitor. We first introduce the notion of *multi-fingerprinting codes*, which is a generalization of fingerprinting codes that allows tracing multiple traitors. (Threshold) traitor tracing based on such codes is a special case of subset-based traitor tracing, and is therefore subject to the lower-bound in Theorem 1. However, we are able to show an even stronger lower-bound for such schemes:

Theorem 2 (Informal). *For a threshold traitor tracing scheme based on multi-fingerprinting codes, that can trace $a \leq t$ traitors, the ciphertext size must scale with $\Omega(t \cdot (n/t)^{a/t})$. In particular, for $a = t$, the ciphertext size must be $\Omega(n)$.*

While this result applies only to fingerprinting codes-based schemes, it gives stronger lower bounds than Theorem 1.

1.2 Our Results: Upper Bounds

Next, we present two generic compilers to construct traitor tracing for general access structures, that can trace to an authorized subset of parties. In particular, for the threshold access structure, we are guaranteed to find t traitors. We then present two concrete instantiations based on our compilers. Lastly, we give a candidate theoretical instantiation based on a new tracing mechanism.

Our Compilers. Our compilers rely on two building blocks: (i) Attribute-based Encryption for general access structures and sufficiently-expressive key policies: this is a generalization of Attribute-based Encryption wherein each party is issued a key associated with a policy and an index, while ciphertexts are labeled with attributes. Decryption succeeds only if a quorum of parties authorized with respect to the access structure comes together, all holding keys whose policies accept the ciphertext's attribute. (ii) Mixed Functional Encryption: this is a special type of functional encryption scheme introduced by [GKW18], wherein the encryption algorithm works with either a public key or master secret key. Ciphertexts encrypted using the public key can only encrypt one type of functionality. On the other hand the secret key encryption can be used to encode many different types of functions, but is only secure as long as the attacker sees a bounded number of such ciphertexts.

Theorem 3 (Informal). *Assume there exists*

- *Mixed Functional Encryption for a special comparison function class, that achieves adaptive security for* $\mathsf{poly}(\lambda)$ *number of secret-key encryption queries and,*
- *adaptively secure attribute-based encryption for general access structures, that can support the policy class of the MFE decryption circuit.*

Then, there exists a traitor tracing scheme for general access structures, which allows tracing to a set of authorized traitors. The ciphertext size scales polynomially with the size of the largest unauthorized subset in the access structure.

Our next compiler only requires adaptive security for the mixed functional encryption scheme wherein the adversary is allowed to make only one secret key encryption query (as opposed to $\mathsf{poly}(\lambda)$ in the previous compiler):

Theorem 4 (Informal). *Assume there exists*

- *Mixed Functional Encryption for a special comparison function class, that achieves adaptive security for* 1*-secret key encryption query and,*
- *adaptively secure attribute-based encryption for general access structures, that can support the policy class of the MFE decryption circuit.*

Then, there exists a traitor tracing scheme for general access structures, which allows tracing to a set of authorized traitors. The ciphertext size scales polynomially with the size of the largest unauthorized subset in the access structure.

Concrete Instantiations. Since all existing constructions of mixed functional encryption and attribute based encryption for general access structures only achieve selective security, we use complexity leveraging to instantiate our compilers. We get the following two results:

Theorem 5. *Under exponential security assumptions, there exists a pairings-based threshold traitor tracing scheme that can trace* t *traitors with ciphertext size* $O(t^2)$*.*

Theorem 6. *Under exponential security assumptions, there exists a lattice-based traitor tracing scheme for DNF access structures that can trace an authorized subset of traitors with ciphertext size* $O(t^2)$*, where* t *denotes the size of the largest unauthorized subset in the access structure.*

Candidate theoretical instantiation. The above positive results for threshold tracing all have ciphertexts growing with t; this is inherent by Theorem 1 for two reasons: they all trace t users, and they all employ universal tracing where the tracing algorithm is independent of the box itself, and in particular independent of the box's run-time.

Our final positive result gives a candidate tracing mechanism based on obfuscation, which traces $a < t$ users by utilizing a non-universal tracing algorithm, and essentially matches Theorem 1:

Theorem 7 (Informal). *For any constant* $c > 0$*, assuming one-way functions and ideal (black box) obfuscation, there exists a threshold traitor tracing scheme whose ciphertext, public key, and secret key sizes are all* $\mathsf{poly}(\lambda)$*, independent of* n *and* t*, which traces* $a = (1 - \lambda^{-c})t$ *users.*

1.3 Additional Related Work

Traceability in Other Primitives. [GSS21] introduced the notion of traceability in secret sharing, wherein a less-than-threshold number of share holders may sell their shares to an adversary. There has been a lot of work on building traceable secret sharing schemes with different efficiency and security tradeoffs [BPR24b, GJP25, Hof24, FG25, FG26]. [BPR25] study traceability in the context of threshold verifiable random functions, wherein a coalition of less-than-threshold parties may sell their VRF evaluation key shares. [DPP20] generalize the notion of traitor tracing to functional encryption.

2 Technical Overview

In this section, we present an overview of our contributions. We start with discussing the models for the lower bounds in Sections 2.1 and 2.2, before presenting our compilers in Section 2.3 and concrete instantiations in Section 2.4 and 2.5. We discuss open problems in Section 2.6.

2.1 Lower Bound on Subset-based Tracing

For our first lower bound, we consider a special case of threshold traitor tracing, which we call subset-based tracing. As we will see later in this section, this captures all existing (threshold) traitor tracing mechanisms that only use the decoder box D as a black-box, which in turn constitute almost all existing traitor tracing methods.

The model. In a subset-based threshold traitor tracing scheme, there is a collection of subsets $S = \{S_x \subseteq [n]\}_{x \in \mathcal{I}}$ where $[n] \in S$. For any subset $S_x \in S$, one can create a ciphertext c so that a set of parties $\mathcal{J} \subset [n]$ can decrypt c if and only if $\mathcal{J} \subseteq S_x$ and $|\mathcal{J}| \geq t$. In other words, for every ciphertext, there is a subset of users S_x that are "authorized" to decrypt that ciphertext. Under normal operations, one encrypts the message μ using the set $[n]$ so that every quorum of t parties can decrypt. However, during tracing, the tracing algorithm will query the decoder D with ciphertexts targeting specific subsets $S_x \in S$.

Intuition for tracing in the subset-based model. In a subset-based traitor tracing scheme, the tracing algorithm tries D on various ciphertexts which "turn off" certain users (namely, those excluded from S_x), and sees if D still manages to decrypt. For example, maybe $t = 3$ and $S_x = \{1, 4, 5, 7\}$. Then any three users in S_x, such as $\{1, 5, 7\}$, will be able to decrypt the ciphertext, while three users not fully contained in S_x, such as $\{1, 4, 10\}$, will not be able to decrypt. Thus, whether or not the decoder is able to decrypt such ciphertexts can reveal information about the set of keys in the decoder's possession.

Applicability to existing tracing schemes. Theorem 1 shows a lower bound on the ciphertext size for subset-based threshold traitor tracing schemes that allow tracing multiple traitors. We now discuss how this lower bound applies to *all* existing tracing techniques:

- Private Linear Broadcast Encryption (PLBE): PLBE was first introduced by [BSW06] in the context of traitor tracing. But recently, many threshold traitor tracing schemes [BPR24a, DDP+25, CMMS25, BHO25] have been constructed based on PLBE techniques. At a high level, in (threshold) traitor tracing based on PLBE, the tracer has the ability (using its secret key) to create ciphertexts associated with intervals, i.e. $S_x = \{i \in [n] : i \leq y\}$ for various values of $y \in \{0, \ldots, n\}$. The tracer then runs the decoder box on such ciphertexts.

– Revocable PLBE: [Zha20, GLW23] use a revocable version of PLBE to achieve non-threshold traitor tracing, and this was adapted to the threshold case in [BPR24a]. While the set system $\{S_x\}$ that can be associated with a ciphertext is more complex than the set of intervals as in standard PLBE, the schemes are an instance of the subset-based tracing paradigm.

– Fingerprinting codes: To understand (threshold) traitor tracing constructions based on fingerprinting codes, let us consider the trivial scheme for $t = 1$ based on binary fingerprinting codes. Herein, we sample two public keys for an encryption scheme for each position of the codeword, corresponding to bits 0 and 1. Then, party i is assigned a subset U_i of the secret keys, based on its codeword: for every position j, it gets the key corresponding to the jth bit of its codeword. A ciphertext encrypts a message to a random position j of the codeword, which can be interpreted as an encryption to a subset C_j of the keys, corresponding to both keys for this position. Observe that for each user, $|U_i \cap C_j| = 1$.
The tracer additionally has the ability to encrypt messages with respect to either the bit 0 or 1 for a position j. Only parties whose jth bit of the codeword matches the bit that the tracer encrypted towards, can decrypt this ciphertext. Such a ciphertext can also be interpret as encrypting to a subset T_j of the keys corresponding to position j. Then, we can define S_x for such a ciphertext as the set of parties i that satisfy $U_i \cap T_j \neq \emptyset$, since only these parties can decrypt such a ciphertext. While we discussed the case of $t = 1$, the above intuition also applies to threshold traitor tracing based on fingerprinting codes as constructed in [BPR24a]. Hence, the fingerprinting-code based tracing is also an instance of the subset-based tracing paradigm.

– Non-black-box Tracing: All existing threshold traitor tracing schemes, including the ones discussed above, only allow the tracer to have black-box access to the decoder. However, in the case of non-threshold traitor tracing, some past works have considered a non-black-box variant wherein the tracing algorithm has access to the the actual code of the decoder. Some early traitor tracing works (e.g. see [KD98, BF99, NP01, TS06, JKL09, ADVW13] and the references therein) worked in this model, but they all made strong assumptions on the decoder – essentially that the decoder took the form of a decryption key. The only prior work which makes non-black-box use of truly arbitrary decoder programs is that of [Zha21]. [Zha21] technically does not fall under the category of subset-based traitor tracing, as it actually utilizes the code of the decoder as opposed to directly making queries. However, the tracing algorithm actually *is* querying the decoder on ciphertexts, just implicitly by performing the query homomorphically. The ciphertexts that are homomorphically queried follow the format of a subset-based traitor tracing scheme. Thus, while [Zha21] does not technically fit the form of a subset-based scheme, it effectively is. In particular, the limits we prove on subset-based schemes will apply to any protocols based on ideas in [Zha21].

Thus, we see that *all* existing tracing mechanisms that do not make any stipulations about the decoder program can be considered subset-based schemes. This includes both the recent threshold schemes, as well as the vast body of work on non-threshold schemes. Hence, our lower bound in Theorem 1 applies quite generally to all such techniques. The precise statement of the lower bound and its proof can be found in Section 4.

2.2 Lower Bound on Fingerprinting-based Tracing

In this section, we focus on tracing based on fingerprinting codes. We start with describing what fingerprinting codes are and how they are useful for tracing.

Background on Traitor Tracing from Fingerprinting codes. A fingerprinting code for alphabet Σ is a set of words where $\Gamma = (w_1, \ldots, w_n)$ wherein each word is of length ℓ, i.e. $w_i \in \Sigma^\ell$. Let us consider $\Sigma = \{0, 1\}$ for now. What makes fingerprinting codes special is that they come equipped with a tracing algorithm Trace that has the following functionality. Say that an adversary is in possession of a subset of words $W \subseteq \Gamma$, and it comes up with a word w^* that is constructed by mixing and matching the bits of the words in W. That is, if for some index $i \in [\ell]$, the ith bit of all words in W is 0, then the ith bit of w^* must be 0 as well (and similarly if the ith bit of all words is 1). We call such a word feasible for W. The guarantee of fingerprinting codes is that the Trace algorithm can trace such a word w^* back to at least one of the words in W.

Equipped with such a code, Boneh and Naor showed how to construct a (non-threshold) traitor tracing scheme from any public key encryption scheme $\mathcal{E}$. The idea is to generate 2ℓ independent key pairs for $\mathcal{E}$ – let us index them using (j, b) for $j \in [\ell], b \in \{0, 1\}$. The overall public key contains all 2ℓ public keys, and the secret key for party i contains one key for each $j \in [\ell]$, depending on the jth position of the word w_i. In particular, party i receives $\{\mathsf{sk}_{j,w_i[j]}\}_{j\in[\ell]}$. To encrypt a message, one chooses a random position $j \leftarrow\!\!\$\ [\ell]$, and encrypts the message under both keys for this position, to get (c_0, c_1).

Tracing relies on the observation that, thanks to the semantic security of $\mathcal{E}$, a coalition of parties holding only $\mathsf{sk}_{j,0}$ cannot distinguish between an honestly generated ciphertext (c_0, c_1) and a ciphertext (c_0, c_1') where c_1' is an encryption to some random message. The same goes for a coalition holding only $\mathsf{sk}_{j,1}$. This observation can be used to extract a word w^* from a pirate decoder D with the property that w^* is indeed feasible for the set of words associated with the corrupted coalition. Hence, we can rely on the tracing algorithm of the fingerprinting code to trace D back to one of the corrupted parties. While the above exposition only applies for the case $t = 1$, [BPR24a] show how to extend this idea to get threshold traitor tracing for arbitrary t using fingerprinting codes. Generally speaking, the ciphertext size scales with the alphabet size $|\Sigma|$, and the secret key size scales with the length of the code ℓ.

The model for lower bound: *multi-Fingerprinting Codes*. The above idea is fundamentally limited to trace only one party, since the Trace algorithm of the code only guarantees to trace one party given a feasible word. If we wish to recover more than one traitors, we need a fingerprinting code capable of identifying multiple users. We first observe that given a single feasible word, it is impossible to trace multiple users, since the word w^* might equal one of the words in W, in which case we can only trace to the party corresponding to that word. Hence, we consider a scenario wherein the cryptography allows us to extract more information from the decoder box, beyond a single feasible word w^*. In particular, we imagine that we extract a subset $\Sigma_j \subseteq \Sigma$ of symbols for each position $j \in [\ell]$ (as opposed to just a single symbol i.e. one bit for each position), with the guarantee that the box must possess at least t codewords whose symbols at that position belong in the subset Σ_j. For instance, if we extract a '1' at a position $j \in [\ell]$, then the box must have t parties whose codeword has a '1' at this position. We refer to the subset of symbols per position as a *multi-word*, and we say that it is feasible for a set of words W if the above guarantee holds with respect to W.

We define a generalization of fingerprinting codes, that we call *multi-fingerprinting codes*, wherein the tracing algorithm takes as input a multi-word that is feasible for a set of words W, and traces it to a subset of W of size t. Our lower bound then proves that for a multi-fingerprinting code, we must either have a large alphabet, or an exponentially-long code. Since an exponentially-long code would lead to exponential-size secret keys, this means that we must have a large alphabet,

hence giving a lower bound on the ciphertext size for schemes based on multi-fingerprinting codes. We refer the reader to Section 5 for more details.

2.3 Upper Bound: Tracing many users by generalizing PLBE

In this section, we describe our generic compilers for building traitor tracing for general access structures. We start with presenting the constructions for threshold, before discussing how to generalize the idea to general access structures. We present our concrete instantiations in Section 2.4.

Starting Point: TTT from TPLBE to trace one party. Our starting point is a generic compiler to build threshold traitor tracing that can trace one party, from a primitive called threshold Private Linear Broadcast Encryption (PLBE), given in [DDP+25]. A threshold PLBE system for n parties has a public key pk, master secret key msk, and secret keys of all parties $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n$. Messages in PLBE can be either encrypted in a *public-key mode* or a *secret-key mode*. The public-key mode encryption (which we also refer to as the normal encryption) requires only the public-key pk and the underlying message, and any t parties can decrypt such public-key mode ciphertexts using their secret keys $\{\mathsf{sk}_j\}$. To encrypt in the secret-key mode, the secret-key encryption algorithm, EncSK, takes as input msk, a message and an index i, and outputs a ciphertext that can only be decrypted by t users with indices greater than i. We call such a ciphertext a secret-key encryption with respect to the set $\{i+1, \ldots, n\}$. In other words, this ciphertext *disables* parties $\leq i$.

Threshold PLBE must satisfy four security properties. The first is the standard notion of semantic security, which says that an adversary that corrupts up to $t-1$ parties, must not learn anything about the encrypted message. The second is *index-hiding*, which requires that an adversary with keys of all parties except i, cannot distinguish between secret-key encryptions to index $i-1$ and i. The third is *message-hiding*, which requires that secret-key encryptions of two different messages m_0, m_1 to index n must be indistinguishable, even if the adversary has all n keys. Lastly, *indistinguishability* requires that a normal encryption of message m created using pk is indistinguishable from a EncSK encryption of m to the index 0, even if the adversary has all n keys.

[DDP+25] show how one can build Threshold Traitor Tracing that allows tracing one corrupt party from Threshold PLBE. For setup, we run the setup algorithm of threshold PLBE, and give secret key sk_i to party i. The master secret key becomes the tracing key. Any message can be encrypted via the public-key mode of threshold PLBE; and this can be decrypted by any t parties. To trace a decoder box D, the tracing algorithm measures (with several samples) the probability that D correctly decrypts a ciphertext encrypted in the secret-key mode to index i for all $i \in [0, n]$. In other words, the tracer tries to *disable* parties one by one and observes the decoder's behavior. If the box D decrypts public-key ciphertexts with probability ϵ, then it must decrypt ciphertexts encrypted to index 0 with roughly the same probability, by indistinguishability. Moreover, D cannot decrypt encryptions to index n, by message-hiding. Hence, there must exist some index $i^* \in [n]$ where D's success probability drops significantly between encryptions to index $i^* - 1$ and i^*. By index-hiding, this gap implies D must contain sk_{i^*}; hence, the tracer simply outputs i^* as a traitor.

Tracing Multiple Parties. Let us now see how we can extend the above framework to trace t parties. Let i^* denote the traitor found via the procedure above. The key observation is that, if we repeat the same procedure as above, i.e. disabling parties one by one, but without ever disabling the party i^* that we already found, then we can catch another traitor! In more detail, say that the tracer has the ability to prepare secret-key encryptions with respect to sets $\{1, \ldots, n\} \cup \{i^*\}$, $\{2, \ldots, n\} \cup \{i^*\}$, and so on till $\emptyset \cup \{i^*\}$ (we will discuss how to extend threshold PLBE to enable this

and the precise security properties later). Then, the tracer can measure the decryption probability of the box D with respect to this series of ciphertexts. Observe that, as before, the box must decrypt ciphertexts encrypted with respect to $\{1,\ldots,n\} \cup \{i^*\}$, if it can decrypt normal ciphertexts. On the other hand, the box cannot decrypt a ciphertext encrypted to $\emptyset \cup \{i^*\}$ – this is because only party i^* is allowed to decrypt such a ciphertext, but it must not be able to decrypt on its own since decryption requires t parties. Hence, there must exist an index $i_2^* \in [n] \setminus \{i^*\}$ where D's success probability jumps significantly between encryptions to $\{i_2^*,\ldots,n\} \cup \{i^*\}$ and $\{i_2^*+1,\ldots,n\} \cup \{i^*\}$. By a similar logic as above, i_2^* must be corrupt. Crucially, we are guaranteed to find a different traitor in this iteration, i.e. i_2^* will not be equal to i^*, because the sets $\{i^*,\ldots,n\} \cup \{i^*\}$ and $\{i^*+1,\ldots,n\} \cup \{i^*\}$ are actually equal – so the decoder cannot distinguish between ciphertexts encrypted to these two sets.

To find t traitors, we can keep repeating the above procedure: the tracer will again prepare a series of ciphertexts, and measure D's behavior, but never disable the traitors found so far i.e. i^*, i_2^*. In more detail, the tracer will encrypt with respect to sets $\{1,\ldots,n\} \cup \{i^*, i_2^*\}$, $\{2,\ldots,n\} \cup \{i^*, i_2^*\}$ and so on till $\emptyset \cup \{i^*, i_2^*\}$. By similar arguments as above, we will again find a new traitor. Hence, the tracer will be able to find t traitors by the end of t iterations[2].

A new primitive: Threshold Private *Set* Broadcast Encryption (TPSBE). To formalize the above intuition, we introduce a new primitive that we call *Threshold Private Set Broadcast Encryption*. This is a generalization of TPLBE, wherein we can encrypt messages in the secret-key mode to an index i and a set $\mathcal{T}$, to produce a ciphertext that can only be decrypted by t users with indices in $\{i+1,\ldots,n\} \cup \mathcal{T}$. In other words, t parties all of which are either $> i$ or in the set $\mathcal{T}$ can decrypt such a ciphertext.

We define four security properties for TPSBE, which generalize those of TPLBE. The first one is semantic security, which is defined exactly as for TPLBE. Next, we have *set-hiding*, which is a generalization of index-hiding. Informally, it requires that an adversary cannot distinguish between encryptions to index-set pairs $(i_0, \mathcal{T}_0)$ and $(i_1, \mathcal{T}_1)$, even if it is given the secret keys of all parties that are either enabled with respect to both encryptions or disabled for both. More formally, the adversary is allowed to get secret keys of all parties in the following set:

$$(\{i_0+1,\ldots,n\} \cup \mathcal{T}_0) \mathbin{\Delta} (\{i_1+1,\ldots,n\} \cup \mathcal{T}_1).$$

Next, we define a generalization of message-hiding, which requires that secret-key encryptions of two different messages to index-set pair $(n, \mathcal{T})$ for any set $\mathcal{T}$ of size $< t$ must be indistinguishable, even if the adversary has all n keys. This captures the following intuition: For a ciphertext encrypted to $(n, \mathcal{T})$, only the parties in $\mathcal{T}$ are *enabled* with respect to this ciphertext. But since $\mathcal{T}$ has less than t parties, this ciphertext cannot be decrypted.

The last property is indistinguishability, which requires that a normal encryption of a message created using pk is indistinguishable from a secret-key encryption with respect to index-set pair $(0, \emptyset)$, even if the adversary has all n keys. Note that, while one could define a more general version of this property wherein we could consider encryptions using pk vs encryptions to $(0, \mathcal{T})$ for arbitrary sets $\mathcal{T}$, the weaker notion we consider above turns out to be sufficient for catching t parties in threshold traitor tracing.

A Subtlety in proving Security. While the idea we described above can be instantiated with a Threshold Private Set Broadcast encryption scheme, a subtlety appears when trying to prove

[2] This approach can be seen as a generalization of the tracing algorithm for the trivial threshold encryption scheme, which also allows tracing t traitors, as observed in [BPR24a].

security. For instance, let us try to prove that the tracing algorithm we described will never blame an honest party by relying on the set-hiding property of threshold PSBE. At a high level, if the tracer blames an honest party h, it means that for some iteration $j \in [t]$ of the tracing algorithm, the decoder box D was able to distinguish between ciphertexts encrypted to $(h-1, \mathcal{T}_j)$ and to $(h, \mathcal{T}_j)$ without ever querying the secret key of party h. Here, $\mathcal{T}_j$ is the set of traitors found by the tracer so far (i.e. in the first $j-1$ iterations of the algorithm). Roughly speaking, we can use such a box D to break set-hiding of the TPSBE scheme.

But, the issue in reducing this to set-hiding is that, the reduction actually does not know which honest party h will the adversary try to blame, or in which iteration j will this occur. Moreover, it also does not know the set $\mathcal{T}_j$, i.e. what traitors would have been found by the jth iteration of the tracing algorithm. One option would be *complexity leveraging*, wherein the reduction simply guesses h and the set $\mathcal{T}_j$ randomly at the beginning of the game. This way, we can directly reduce to set-hiding, but this will lead to a factor of $\binom{n}{t-1}$ loss in security!

Alternatively, the reduction can guess h randomly from $[n]$, guess iteration j randomly from $[t]$, and then, once it receives the decoder box D from the adversary, it can simulate the first $j-1$ iterations of the tracing algorithm, to determine the set $\mathcal{T}_j$. But crucially, this means that (i) the reduction will determine the challenge sets $(i_0, \mathcal{T}_0) = (h-1, \mathcal{T}_j)$, $(i_1, \mathcal{T}_1) = (h, \mathcal{T}_j)$ *adaptively*, i.e. after the adversary has already queried secret keys of parties of its choice. Moreover, (ii) the reduction will need to make a large number of secret key encryption queries to its set-hiding challenger, i.e. EncSK queries, to even run the $j-1$ iterations of the tracing algorithm (recall each iteration requires encrypting to arbitrary index-set pairs and measuring the decoder's behavior on those encryptions). This means, we need to rely on adaptive, $\mathsf{poly}(\lambda)$-EncSK query set-hiding of threshold PSBE to prove that honest parties are never blamed. For a similar reason, we will also need to rely on adaptive, $\mathsf{poly}(\lambda)$-EncSK query message-hiding of threshold PSBE to prove that t traitors are found. Note that this issue does not appear in prior TTT constructions that only trace to one party – since they only run the first iteration of the tracing algorithm, the reduction simply needs to guess the honest party h that the adversary will try to blame, which only leads to a $1/n$ security loss.

Unfortunately, constructing threshold PSBE that achieves these strong notions of security, i.e. which allow adaptivity and $\mathsf{poly}(\lambda)$ number of EncSK queries, turns out to be a difficult task, as we will see later in this section. Hence, we present a second compiler that also builds TTT from TPSBE, but relies on weaker security notions.

Our Second Compiler. Our second compiler only requires threshold PSBE with adaptive set-hiding and message-hiding (without any EncSK queries) to build TTT. We still rely on the same key idea as above, wherein we do a total of t iterations, and trace one traitor in each. But now, we will instantiate t copies of threshold PSBE and use a different copy for each iteration. In more detail, we run the setup of TPSBE t times. The overall keys are the concatenation of the keys for all t copies, for instance, $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_t)$. To encrypt a message, we randomly sample an instance $j \in [t]$, and encrypt using pk_j.

Now, for tracing, let us first consider a 'perfect' box, that successfully decrypts all ciphertexts. This means the box can decrypt a ciphertext encrypted to any of the t TPSBE instances. Then, for tracing, we can use the first TPSBE instance for the first iteration, i.e. we measure the decryption success of D on ciphertexts encrypted to instance 1 and index-set pairs $(0, \emptyset), \ldots, (n, \emptyset)$. Roughly speaking, by security of the first TPSBE instance, we are guaranteed to find a traitor i^* in this iteration. Then, for the second iteration, we will rely on the second TPSBE instance, i.e. we will

use ciphertexts encrypted to instance 2, and index-set pairs $(0, \{i^*\}), \ldots, (n, \{i^*\})$. Again, we are guaranteed to find a different traitor i^* by a similar reasoning as above. We can keep going and find t traitors in t iterations.

Let us now see how we can prove security with just adaptive set-hiding (and message-hiding) of TPSBE. Again, let us focus on proving that the adversary can never blame an honest party, by relying on adaptive set-hiding of TPSBE. As before, we know that if the adversary can blame some party h, then for some iteration $j \in [t]$ of the tracing algorithm, the box must be able to distinguish between ciphertexts encrypted for instance j, with respect to $(h-1, \mathcal{T}_j)$ and to $(h, \mathcal{T}_j)$ where $\mathcal{T}_j$ is the set of traitors found in the first $j-1$ iterations of the tracing algorithm. The reduction can guess h and j randomly from $[n]$ and $[t]$ respectively as before. We will reduce to set-hiding of the jth instance, i.e. the reduction will set the jth public key to be the challenge public key. It can then run the Setup of all the remaining instances $j' \neq j$ on its own. Lastly, to determine the set $\mathcal{T}_j$, the reduction needs to simulate the first $j-1$ iterations of the tracing algorithm. But the key observation here is that the first $j-1$ iterations are executed with respect to the first $j-1$ instances of TPSBE. And since the reduction ran the setup for all these instances, it can compute any EncSK encryption for these instances on its own, and does not need to query its set-hiding challenger which is only relevant for the jth TPSBE instance! Hence, we can reduce to adaptive (0-EncSK query) set-hiding of TPSBE with only a polynomial security loss. By a similar reasoning, we will be able to prove full traceability with indistinguishability, adaptive set-hiding and message hiding. We note that this compiler introduces a multiplicative overhead of t in the public and secret keys, but the ciphertext only grows by an additive factor of $\log_2(t)$.

So far, we have only discussed how to trace perfect boxes. But, the adversary could make a box that only works for certain values of j. In the extreme case, the box may only decrypt ciphertexts encrypted with respect to one particular TPSBE instance, in which case we will only be able to trace one traitor. To get around this issue, we instantiate many instances of TPSBE (in particular, $O(t)$ instances), and only focus on boxes which work with a *large probability*. For such boxes, we show that it must be able to decrypt ciphertexts for many TPSBE instances. We set the parameters such that this holds for t instances, which are enough for us to find a set of t traitors.

This technique results in *Limited* threshold traitor tracing, which means that we are guaranteed to find t traitors only for boxes that work with high probability[3]. But luckily, prior works [DDP+25] have shown how to generically transform limited threshold traitor tracing (that trace one traitor) to threshold traitor tracing that can trace one traitor for boxes with arbitrary success probability, with only a $\mathsf{poly}(\lambda)$ overhead. In Section 9.2, we show that this compiler also applies to our setting where we trace t parties. Hence, by combining the two compilers, we get threshold traitor tracing that can trace t parties from a TPSBE with adaptive set-hiding and message-hiding.

Constructing Threshold Private Set Broadcast Encryption from ABTE and MFE. Let us now discuss how we can construct threshold PSBE, to instantiate the above compilers.

Starting Point: Threshold PLBE from ABTE and MFE. We first review how we can construct threshold PLBE from Attribute based Threshold Encryption (ABTE) and Mixed Functional Encryption (MFE), as shown in [DDP+25]. We start with defining these building blocks below:

In a t-out-of-n (key-policy) ABTE, each user's secret key is tied both to a policy and to the user's index. Specifically, the key for user i and policy $C_i : \mathcal{X} \to \{0,1\}$ is denoted sk_{i,C_i}. As in standard ABE, encryption of a message m is associated with a public attribute $x \in \mathcal{X}$. Such an

[3] Note that in the traitor tracing literature, this is traditionally referred to as 'threshold' traitor tracing, but we use 'Limited' to avoid confusion with threshold decryption, following [DDP+25].

encryption can only be decrypted by a coalition of t parties whose policies are all satisfied by x. Concretely, each party can only compute a partial decryption share using its key, and the message can be recovered only by combining t such shares from parties with satisfying policies i.e. $C_i(x) = 1$ for all t parties i. ABTE must satisfy semantic security, which requires that an adversary cannot learn anything about a message encrypted with attribute x^*, even if it can query (i) an unbounded number of keys for parties with policies not satisfied by x^* (as in ABE security), (ii) keys of up to $t-1$ parties with satisfying policies, and (iii) decryption shares for valid ciphertexts of its choice (as in threshold encryption security).

The second building block is mixed functional encryption, which supports two encryption modes: (i) public-key (or normal) encryption, which takes only the public key as input and outputs a ciphertext ct, and (ii) secret-key encryption, which takes the master secret msk and a function f to produce a ciphertext ct_f. Decryption keys are tied to messages: a normal ciphertext ct will decrypt to 1 when using the key sk_m for any message m. Additionally, decrypting a secret-key ciphertext ct_f using sk_m will output $f(m)$. MFE security requires (a) *accept-indistinguishability*, which states that normal ciphertexts are indistinguishable from secret-key ciphertexts for a function f, unless one has a secret key for message m such that $f(m) \neq 1$, and (b) *function-indistinguishability*, which requires that an adversary learns nothing more about a function f except for the values $f(m)$ for messages m whose keys are in its possession.

To build threshold PLBE from ABTE and MFE, we run setup for both the primitives. To encrypt a message, we encrypt using ABTE, with an MFE ciphertext as the attribute. In the public-key mode, the attribute is a normal MFE ciphertext ct. To encrypt to index i in the secret-key mode (recall we only encrypt to an index in threshold PLBE), we set the ABE attribute to $\mathsf{ct}_{\mathsf{comp}_i}$, an MFE secret-key ciphertext encrypting the comparison function comp_i that outputs 1 only if the input is $> i$. Each party j receives an ABE secret key with the policy being the MFE decryption circuit, hard-wired with the MFE secret key sk_j corresponding to index j. Clearly, any t parties can decrypt a normal ciphertext since the policy for any key evaluates to one on ct, by MFE correctness. On the other hand, decrypting $\mathsf{ct}_{\mathsf{comp}_i}$ only outputs one for parties $j > i$, so only t such parties can decrypt the ciphertext encrypted to i, as desired. One can show semantic security of TPLBE by relying on semantic security of ABTE. Index hiding of TPLBE can be reduced to function-indistinguishability of the MFE. Message hiding can be reduced to ABTE semantic security, and indistinguishability can be reduced to accept-indistinguishability of MFE. Crucially, we only need the MFE to support the comparison function for building threshold PLBE.

Constructing Threshold PSBE. To construct threshold PSBE, we follow the same approach as above. The only difference is how we encrypt to index-set pairs in the secret-key mode: To encrypt to $(i, \mathcal{T})$, we encrypt via ABTE with the attribute equal to $\mathsf{ct}_{\mathsf{comp}_{i,\mathcal{T}}}$, an MFE secret-key ciphertext encrypting the *special comparison function* $\mathsf{comp}_{i,\mathcal{T}}$, which outputs 1 only if the input is $> i$ or $\in \mathcal{T}$. The rest of the compiler remains the same as above. Note that we do need MFE to support special comparison functions.

The security proofs for TPLBE can also be generalized to prove semantic security, set-hiding, message-hiding and indistinguishability for TPSBE. But for the TPSBE to achieve adaptive $\mathsf{poly}(\lambda)$-query message-hiding and set-hiding, we need the underlying ABTE to satisfy adaptive semantic security, and the MFE to satisfy adaptive $\mathsf{poly}(\lambda)$-query function-indistinguishability, respectively. In particular, for the MFE, the adversary must be allowed to query polynomially-many secret key ciphertexts for functions of its choice and then choose the challenge functions adaptively, in the function-indistinguishability game.

Combining this compiler with our first TTT compiler results in Theorem 3 for the threshold access structure (we discuss general access structures below). Due to some subtleties, even for our second compiler which requires only adaptive security for the TPSBE, we still need MFE with 1-query adaptive security. This results in Theorem 4. We refer the reader to the main body for more details.

Extending to General Access Structures. So far we have shown how to construct Threshold Traitor Tracing where we can trace t traitors. For access structures beyond threshold, we first start with presenting the definition for traitor tracing for general access structures. Herein, decryption will require a subset of parties that is authorized with respect to the access structure. In the semantic security game, the adversary will be allowed to query the secret keys of any unauthorized subset of parties of its choice. For tracing, we require the tracer to trace a decoder box to *an authorized subset of traitors* – note that we cannot put any constraints on the number of traitors found, since the size of different authorized sets may vary greatly for some access structures.

We now show that our compilers almost directly generalize to construct traitor tracing for general access structures. We start with defining Private Set Broadcast Encryption for general access structures (PSBEGen). The definition is a generalization of that for threshold PSBE, but decryption will require a subset of parties that is authorized with respect to the access structure. In the semantic security game, the adversary will be allowed to query the secret keys of any unauthorized subset of parties of its choice. The rest of the definitions remain the same. It is easy to see that the same approach as in our first compiler directly generalizes to this scenario: the tracer can find one party in each iteration. The only difference would be that the tracer may need to keep iterating until it finds an authorized subset of traitors. To be precise, the number of iterations will be upper bounded by the size of the largest unauthorized subset (plus one). For our second compiler – the key idea remains the same, but we now need to scale the number of PSBEG instances proportionally to the size of the largest unauthorized subset (plus one), to ensure we can indeed trace to an authorized subset of parties.

Lastly, we also show how to build PSBEGen from a MFE and a new primitive, attribute-based encryption for general access structures, which is a generalization of ABTE. This compiler is also a direct generalization of the one described above for building threshold PSBE.

2.4 Instantiating the Compilers

We now discuss how to instantiate our compilers. Unfortunately, the only known MFE that achieves adaptive security, even with 1 secret-key encryption query, relies on indistinguishability obfuscation or iO. Moreover, there are only two known constructions of Attribute based encryption for certain access structures, and both of them only achieve selective security. Hence, we rely on complexity leveraging to instantiate our compilers.

Threshold Traitor Tracing from Pairings [DDP+25] construct an Attribute based Threshold encryption scheme that achieves selective semantic security based on Decisional Bilinear Diffie Hellman (DBDH) assumption. Additionally, [Zha25] presented a MFE scheme from pairings that also relies on DBDH assumption, and achieves selective security. We observe that this MFE scheme

[4] We use Limited to denote that the tracer can only trace boxes which can decrypt with high (but constant) probability. This is traditionally referred to as 'threshold' traitor tracing (see [Zha25]), but we use Limited, following [DDP+25].

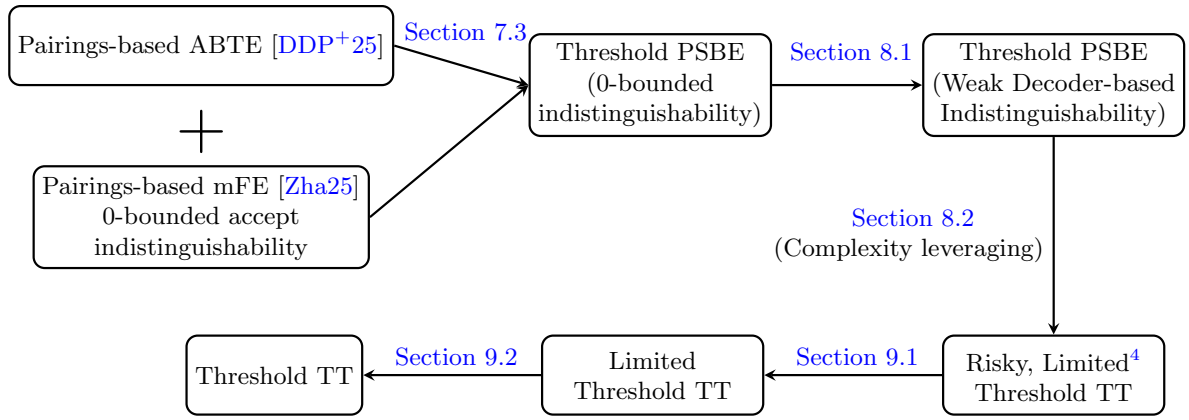


Fig. 1. The sequence of compilers for constructing our pairings-based TTT scheme which traces to t traitors. The definitions of 0-bounded accept indistinguishability for MFE, q-bounded indistinguishability, Weak Decoder-based indistinguishability for TPSBE, and risky, Limited TTT can be found in Sections 6.1, 6.2, 7.2 and 8 respectively. Note we need to rely on complexity leveraging since the threshold PSBE only achieves selective set-hiding and message-hiding.

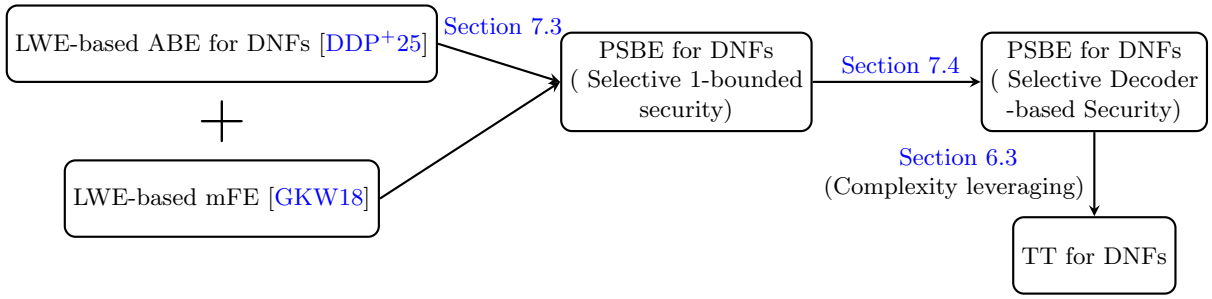


Fig. 2. The sequence of compilers for constructing our LWE-based TT scheme for DNFs, wherein we can trace to an authorized subset of traitors. The q-bounded and decoder-based notions of security for PSBE for general access structures are formalized in Section 6.2.

can support the special comparison function (as defined in Section 2.3), with a ciphertext of size $\Omega(t)$. By plugging these two building blocks into our TPSBE compiler, we get a selectively secure TPSBE with ciphertext size $\Omega(t)$. Then, by assuming exponential security of this TPSBE scheme, and instantiating our first TTT compiler with this scheme, we get a Threshold Traitor tracing construction that can trace t traitors, with ciphertext size $O(t^2)$.

Unfortunately, Zhandry's MFE only satisfies a weak notion of security. Consequently, the resulting TPSBE after applying the first compiler also only achieves weak security – unfortunately, the TPSBE to TTT compiler cannot be applied to such a TPSBE scheme. To solve this issue, we observe that [DDP+25] also deal with the same issue to achieve their optimal TTT constructions. They construct a different sequence of compilers that allow constructing TTT from an MFE with weak security (with the same key ideas as the TPLBE to TTT compilers described above). We show that the same sequence of compilers can also be applied in our setting to achieve TTT that traces t parties starting from MFE with weak security and ABTE. Figure 1 shows the full sequence of compilers. This recovers our first concrete result stated in Theorem 5.

Traitor Tracing for DNFs from Lattices [DDP+25] present a generic compiler based on LWE, that constructs Attribute-based Encryption for any access structure, given a secret sharing scheme with certain properties for the same access structure. We observe that we can instantiate this compiler with the linear secret sharing scheme for DNFs constructed in [LW11][5], to get ABE for DNFs. We can then plug this along with the lattice-based MFE constructed in [GKW18] into our compilers to get a LWE-based PSBE scheme for DNFs with ciphertext size $O(t)$, where t is the size of the largest unauthorized set for the access structure (plus one). Assuming exponential security of this PSBE for DNFs, we can plug this into our compiler to get a Traitor Tracing scheme for DNFs, with ciphertext size $O(t^2)$. Figure 2 shows this series of compilers. This recovers our second concrete result stated in Theorem 6.

2.5 A Candidate for tracing many users with optimal parameters

Here, we very briefly explain how we achieve Theorem 7. The tracing algorithms above followed a natural blueprint of trying to turn off all users but the accused users. It is not hard to see that, since the set of accused users can be truly arbitrary, and the ciphertext must (information-theoretically) encode the set of accused/turned-on users, that ciphertexts must grow with the number of accused users. However, for a number of accused users a only somewhat smaller than t, namely $a = (1-\lambda^{-c})t$, the lower bound in Theorem 1 becomes vacuous.

Our idea to trace a users is, very roughly, to keep the accused users turned on, but only turn off a small but random subset of the other users. Of course, even describing a truly random small set of users to turn off will require large ciphertexts. But here, since the set is random, we can *pseudorandomly* sample which users to turn off. In fact, we will pseudorandomly sample the turned off users from among *all* possible users, including the users already accused, and then simply sample again in the event that any of the accused users fail to be turned on. This allows us to describe the set of turned on/off users succinctly using a *pseudorandom function*. The key to efficiency is ensuring that the number trials to produce a good sample remains polynomial. It turns out that the number of samples remains efficient for $a = (1 - \lambda^{-c})t$ for a constant c, and the lower-bound in Theorem 1 shows that we cannot do any better.

To actually implement this idea, we employ candidate cryptographic obfuscation. To justify security, we model the obfuscated program as an ideal black-box, similar to the notion of virtual-black box (VBB) obfuscation. While ideal/VBB obfuscation are known to be impossible for general programs [BGI+01], the counterexamples are contrived and involve self-referential programs. Our programs do not have this flavor, and so it is plausible that our scheme is secure despite the impossibilities. There are notable barriers to proving security under a standard obfuscation notion such as indistinguishability obfuscation (iO). We give our full protocol and discuss these barriers in Section 10.

2.6 Open Problems

Our work leaves open several interesting questions for future research: The first one is constructing Attribute-based Threshold Encryption with adaptive security. Such a primitive would allow instantiating our compilers with MFE based on indistinguishability obfuscation (iO), which achieves adaptive $\mathsf{poly}(\lambda)$-query security. Note that constructing adaptively-secure ABTE seems to be a

[5] This secret sharing scheme was also used in the context of silent encryption for DNFs from lattices in [CW26].

non-trivial task because this primitive is stronger than both ABE and threshold encryption. In particular, even though there have been works on achieving adaptive security for ABE, they can only handle key queries for policies not satisfied by the challenge attribute, albeit adaptively. But in the adaptive security game for ABTE, the adversary could additionally query up to $t-1$ keys for policies satisfied by the challenge attribute. This means that the reduction would not know whether a key query sent by the adversary is for a policy satisfied or unsatisfied by the challenge attribute, since the adversary can choose the challenge attribute adaptively. This makes the task of achieving adaptivity much more challenging.

Second, the only known construction for MFE that achieves adaptive security is using indistinguishability obfuscation or iO. We leave the task of designing adaptively secure MFE from weaker assumptions, even with 1-query security, as an interesting problem. Combining this with an adaptively secure ABTE using our compilers would result in more efficient threshold traitor tracing schemes.

Third, a natural question is to prove security of our obfuscation-based candidate under standard notions like indistinguishability obfuscation, or develop new provable schemes from other tools, beyond fingerprinting-code or subset-based, for tracing close to t users with $\mathsf{poly}(\lambda)$-sized ciphertexts.

2.7 Outline

We define threshold traitor tracing that allows tracing many traitors in Section 3. We present our subset-based and fingerprinting-code based lower bounds in Sections 4 and 5 respectively. We present our compilers in Sections 6 to 9, and our candidate theoretical construction in Section 10.

3 Definitions

We start with defining threshold traitor tracing in this section. We will generalize this definition to general access structures in Section 6.1. A threshold traitor tracing scheme is a tuple of five polynomial-time algorithms $\mathsf{TTT} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine}, \mathsf{Trace})$:

- $\mathsf{KeyGen}(1^\lambda, 1^n, 1^t, 1^a) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{tk})$ is the probabilistic key generation algorithm. It takes in the security parameter $\lambda \in \mathbb{N}$, the number n of decryptors, the threshold t, and the minimal number of accused users a. It outputs a public key pk, secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$, and the tracing key tk.
- $\mathsf{Enc}(\mathsf{pk}, \mu) \to \mathsf{ct}$ is the probabilistic encryption algorithm. It takes as input the public key pk and a message μ, and outputs a ciphertext ct. We may use $\mathsf{Enc}(\mathsf{pk}, \mu; r)$ to refer to a deterministic encryption algorithm, which takes as input a randomness r sampled from a randomness space $\mathcal{R}$.
- $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}) \to d_i$ is the deterministic decryption algorithm. It takes in a decryptor's secret key sk_i and a ciphertext ct, and its output is a decryption share d_i of ct under sk_i.
- $\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in\mathcal{J}}) \to \mu/\perp$ is the deterministic combiner algorithm. Its input is the public key pk, a ciphertext ct, a subset $\mathcal{J}$ of $[n]$, and decryption shares $\{d_j\}_{j\in\mathcal{J}}$. It outputs either a message μ or a rejection symbol $\perp$.
- $\mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^{1/\epsilon}, \mu_0, \mu_1) \to \mathcal{J}$ is the probabilistic tracing algorithm that takes as input the public key pk, the tracing key tk, an error bound $\epsilon = \epsilon(\lambda)$ and messages μ_0, μ_1. Trace also has oracle access to a *"decoder"* algorithm D. It outputs a subset $\mathcal{J} \subseteq [n]$.

Correctness. An honestly generated ciphertext should be correctly decrypted by any subset of t decryptors. We formalize this notion in Definition 1 below.

Definition 1. *A threshold traitor tracing scheme* TTT *is said to be correct if, there exists a negligible function* $\mathsf{negl}(\lambda)$ *of the security parameter* λ*, such that, for all* $\lambda \in \mathbb{N}$*, all* $0 < a \leq t < n \in \mathbb{N}$*, all* t*-sized subsets* $\mathcal{J} \subseteq [n]$*, and all messages* μ *in the message space, it holds that*

$$\Pr\left[\mu' = \mu \ : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{tk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, 1^n, 1^t, 1^a) \\ c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mu) \\ \forall\ i \in \mathcal{J},\ d_i \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}) \\ \mu' \leftarrow \mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j \in \mathcal{J}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Semantic Security. We require that a threshold traitor tracing scheme must satisfy the notion of semantic security, as is standard for threshold decryption. Roughly speaking, we require that an adversary must not be able to learn anything about the encrypted message, even if it can corrupt upto $t-1$ parties. We also allow the adversary to query partial decryptions of *valid* ciphertexts of its choice. We use the formal definition put forth by [DDP+25]. We consider the semi-adaptive notion wherein the adversary must make all the key queries before it makes any partial decryption query.

Definition 2 (CPA security). *We say that* TTT *satisfies semi-adaptive CPA security if for every probabilistic polynomial time adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{sa-cpa}}_{\mathcal{A},\mathsf{TTT}}(\lambda) := \left|\frac{1}{2} - \Pr[\mathbf{IND\text{-}CPA}^{sa}_{\mathcal{A},\mathsf{TTT}}(\lambda) = 1]\right|.$$

Experiment $\mathbf{IND\text{-}CPA}_{\mathcal{A},\mathsf{TTT}}(\lambda)$

1 : $\mathcal{J} \leftarrow \emptyset$
2 : $(n, t, a, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
3 : $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{tk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, 1^n, 1^t, 1^a)$
4 : $(\mu_0, \mu_1, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}^{\mathsf{sk}(\cdot), \mathsf{dec}(\cdot,\cdot,\cdot)}(\mathsf{state}, \mathsf{pk}, \mathsf{tk})$
5 : $b \leftarrow\!\$\ \{0,1\},\ \mathsf{ct} \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mu_b)$
6 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{sk}(\cdot), \mathsf{dec}(\cdot,\cdot,\cdot)}(\mathsf{state}, \mathsf{ct})$
7 : **if** $|\mathcal{J}| \geq t$ **then return** 0
8 : **return** $b' = b$

Oracle $\mathsf{sk}(i)$

1 : **if** $i \notin [n]$ **then return** $\perp$
2 : $\mathcal{J} \leftarrow \mathcal{J} \cup \{i\}$
3 : **return** sk_i

Oracle $\mathsf{dec}(\mu, r, i)$

1 : **if** $i \notin [n]$ **then return** $\perp$
2 : **return** $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(\mathsf{pk}, \mu; r))$

Fig. 3. The semantic security experiment for a threshold traitor tracing scheme TTT and an adversary $\mathcal{A}$.

Traceability. Informally, if the decoder D, given a ciphertext c, can learn any information about the encrypted message, then Trace traces D and accuses a subset of the coalition that "manufactured" it. In particular, we want to guarantee at least a users are accused for any successful decoder. Moreover, it should not falsely accuse any honest party. This is formalized by the following definition:

Definition 3. *Let $\epsilon = \epsilon(\lambda)$ and $\delta = \delta(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A threshold traitor-tracing scheme $\mathsf{TTT} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine}, \mathsf{Trace})$ is (a, ϵ, δ)-secure if for every PPT adversary $\mathcal{A}$ and for every $\lambda \in \mathbb{N}$, the following holds. Consider the following experiment:*

- $(1^n, 1^t, 1^a, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
- $(\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_n, \mathsf{tk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, 1^n, 1^t, 1^a)$
- $(D, \mu_0, \mu_1) \leftarrow\!\$\ \mathcal{A}^{\mathsf{sk}(\cdot)}(\mathsf{state}, \mathsf{pk})$*, where* $\mathsf{sk}(i)$ *returns* sk_i*. Let $\mathcal{J}$ be the set of* $\mathsf{sk}(i)$ *queries made by $\mathcal{A}$.*
- *Let* $\mathcal{J}' \leftarrow\!\$\ \mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^{1/\epsilon(\lambda)}, \mu_0, \mu_1)$.
- *Define:*
  - $\mathsf{GoodDec}$ *is the event that* $\Pr[D(c) = b : b \leftarrow\!\$\ \{0,1\}, c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mu_b)] \geq 1/2 + \epsilon(\lambda)$
  - GoodTr *is the event that* $|\mathcal{J}'| \geq a$
  - BadTr *is the event that* $\mathcal{J}' \not\subseteq \mathcal{J}$.
- *Then* $\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \delta(\lambda)$ *and* $\Pr[\mathsf{BadTr}] \leq \delta(\lambda)$.

We also refer to $\delta(\lambda)$ as the advantage of $\mathcal{A}$ i.e. $\mathsf{Adv}^{\mathrm{trac}}_{\mathcal{A},\mathsf{TTT}}(\lambda) := \delta(\lambda)$.

Variations. We can also place various restrictions on the parameters n, t, a for which a traitor tracing protocol operates.

Plain (non-threshold) traitor tracing. The original non-threshold version of traitor tracing corresponds to schemes that work for $a = t = 1$.

Existing threshold traitor tracing. All existing work on threshold traitor tracing with sublinear ciphertexts only works for $a = 1$; with the exception of [BFO26], which can trace $\Omega(n)$ traitors with ciphertexts of size $\sqrt{n}$, but their results are in a non-standard model where correctness is only required to hold for a random set of t parties. This is in contrast to the standard notion of correctness which requires decryption to work for every set of t parties.

However, it has been observed in [BFO26] that threshold traitor tracing schemes that trace only one party can actually be built generically by independently running both a threshold (non-traitor tracing) scheme and a plain traitor tracing (non-threshold) scheme, and 2-out-of-2 secret sharing the message to the two schemes. Moreover, a number of limitations have been observed with such a definition [BFO26]. Therefore, we seek to obtain threshold traitor tracing where $a > 1$, ideally $a = t$, while achieving standard correctness.

4 Lower-bound on Subset-based Traitor Tracing

We start with formally defining subset-based tracing before presenting our lower bound:

In a subset-based threshold traitor tracing scheme, for each public key pk, there is a collection $S = (S_x)_{x \in \mathcal{I}}$ of subsets $S_x \subseteq [n]$. Then the tracing algorithm $\mathsf{Trace}^D(\mathsf{pk}, \mathsf{tk}, 1^{1/\epsilon(\lambda)}, \mu_0, \mu_1)$ will satisfy the following guarantee: each ciphertext c that Trace queries to D will be associated with some $S_x \in S$ and a message μ, such that, for any subset $\mathcal{J} \subseteq [n]$ of size t, $\mathsf{Combine}(\mathsf{pk}, c, \{d_j\}_{j \in \mathcal{J}}) = \mu$

if and only if $\mathcal{J} \subseteq S_x$, where $d_i = \mathsf{Dec}(\mathsf{sk}_i, c)\ \forall\ i$. In other words, for every ciphertext, there is a subset of the users which are "authorized", and the honest decryption algorithm only succeeds in outputting μ if *all* of the decrypting users are authorized.

Theorem 8. *Let* TTT *be a subset-based* t*-out-of-*n *threshold traitor tracing scheme that is* (a, ϵ, δ)*-secure for any negligible* δ*,* $a \leq t$ *and any* $\epsilon \geq 0$*. Then we obtain the following lower-bounds on the ciphertext size* $|\mathsf{ct}|$*:*

- *Assume* $1 < a \leq t - \omega(1)$ *and* $n \geq \lambda t(t - a + 2) + 1$*. Then* $|\mathsf{ct}| \geq \frac{a-1}{t-a+1} \times \omega(\log \lambda)$*.*
- *Assume* $a \geq t - O(1)$*. Then* $|\mathsf{ct}| \geq \Omega(t \log \lambda)$

Proof. We can assume without loss of generality that, for each S_x in the subset collection $S = \{S_x\}_{x \in \mathcal{I}}$, there is some ciphertext ct associated to S_x. This is because, if there is an S_x without an associated ct, we can simply remove S_x from S and TTT will still be subset-based for the new, smaller, collection of sets. This allows us to lower bound the size of ciphertexts in terms of the size of the index set: $|\mathsf{ct}| \geq \log_2 |\mathcal{I}|$. Our proof works by lower-bounding the size of $\mathcal{I}$.

Define $v \in [t(t - a + 2), n - 1]$ and

$$Q = \lambda \times \binom{n-a}{t-a+1} \Big/ \binom{v-a+1}{t-a+1}$$

Observe that if $v = n - 1$ then $Q = \lambda$, and decreasing v results in a larger Q. The next lemma bounds how Q can grow as v shrinks.

Lemma 1. $Q \in \lambda((n-1)/v)^{t-a+1} \times [1, e]$

Proof (of Lemma 1). For the lower-bound, by the definition of Q,

$$Q/\lambda = \frac{\binom{n-a}{t-a+1}}{\binom{v-a+1}{t-a+1}} = \frac{(n-1-a+1)(n-1-a)\cdots(n-1-t+1)}{(v-a+1)(v-a)\cdots(v-t+1)} \geq \left(\frac{n-1}{v}\right)^{t-a+1}$$

For the upper-bound, we have

$$\begin{aligned} Q/\lambda &= \frac{(n-1-a+1)(n-1-a)\cdots(n-1-t+1)}{(v-a+1)(v-a)\cdots(v-t+1)} \leq \left(\frac{n-1-t}{v-t}\right)^{t-a+1} \\ &= \left(\frac{n-1}{v}\right)^{t-a+1} \times \left(1 + \frac{(n-1-v)t}{(v-t)(n-1)}\right)^{t-a+1} \end{aligned}$$

Now we argue that $(n-1-v)t/(v-t)(n-1) \leq 1/(t-a+1)$. Indeed,

$$\frac{1}{t-a+1} - \frac{(n-1-v)t}{(v-t)(n-1)} = \frac{(n-1)(v-t(t-a+2)) + vt(t-a+1)}{(t-a+1)(v-t)(n-1)} \geq 0$$

where the latter inequality comes from the assumption that $v \geq t(t-a+2)$, $n-1 \geq v \geq 0$, and $t \geq a$. Therefore, we have

$$Q/\lambda \leq \left(\frac{n-1}{v}\right)^{t-a+1} \times \left(1 + \frac{1}{t-a+1}\right)^{t-a+1} \leq \left(\frac{n-1}{v}\right)^{t-a+1} \times e$$

□

Ultimately, Q will correspond to the running-time of our adversary, and so it must be polynomial. By our choice of $(n-1) \geq \lambda t(t-a+2)$, we can take n as small as v or as large as (at least) λv. For $a \in [t - O(1), t]$, Q/λ ranges from $O(1)$ to $\Omega(\lambda)$ as v ranges between its lower- and upper-bounds. For $a = t - \omega(1)$, λ^{t-a+1} is super-polynomial, which allows us to take Q to be an arbitrarily large polynomial by choosing appropriate $(n-1)/v$.

Consider the following adversary $\mathcal{A}$, which depends on v (or equivalently Q):

- First, choose a random subset $K \subseteq [n]$ of size $a-1$. This will be the "sacrificed" set of users, which $\mathcal{A}$ allows to be accused.
- Choose a random $i^* \in [n] \setminus K$. $\mathcal{A}$ will query the secret keys sk_i for all $i \neq i^*$.
- $\mathcal{A}$ will then choose two random messages μ_0, μ_1 and output the decoder D, where $D(\mathsf{ct})$ does the following:
  - For each $i \in [n] \setminus \{i^*\}$, run $d_i \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$
  - Repeat the following Q times: choose a random set $\mathcal{J}$ of size t such that $\mathcal{J} \subseteq [n] \setminus \{i^*\}$ but $K \subseteq \mathcal{J}$; run $\mu' \leftarrow \mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \mathcal{J}, \{d_j\}_{j \in \mathcal{J}})$; If $\mu' = \mu_b$ for some b, abort and output b.
  - If D has not yet aborted after Q trials, output a random bit.

In other words, D tries decrypting with numerous random subsets $\mathcal{J}$ that are larger than K. If any of them successfully decrypt, D will output that result. Otherwise, D outputs a random bit.

What we will show is that, unless Trace can query D on ciphertexts corresponding to authorized user sets S_x that are "close" to K, then Trace will be unable to learn any information about i^*, other than the fact that it is outside of K. Here, "close" means that S_x contains K but is not too much larger than K. But without learning any information about i^*, Trace can't accuse any user outside of K, since if it does, it risks accusing an honest user. Since K only contains $a-1$ users, Trace is therefore unable to accuse a users. The ciphertext lower-bound then comes from the fact that a small number of S_x cannot be "close" to very many possible sets K. Thus, in order to trace with good probability, it is necessary for there to be many S_x, which in turn means the number of possible ciphertexts is large. We now give the proof in detail.

Let i be the smallest user outside of K that is accused by Trace; if no such user exists, set $i = \perp$. First, we observe that D will always decrypt valid ciphertexts $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \mu_b)$. Therefore, $\Pr[\mathsf{GoodDec}] = 1$, and so $\Pr[\mathsf{GoodTr}] \geq 1 - \delta(\lambda)$. In other words, Trace must accuse a users with probability $1 - \delta(\lambda)$, which in particular means $\Pr[i = \perp] \leq \delta(\lambda)$.

Next, $\Pr[\mathsf{BadTr}] \leq \delta(\lambda)$, which means $\Pr[i = i^*] \leq \delta(\lambda)$. We now claim that i^* is actually statistically hidden to Trace. Consider a query ct to D, associated with an authorized user set S_x. We break into several possibilities:

- Case 1: $K \not\subseteq S_x$.
- Case 2: $K \subseteq S_x$, but $|S_x| > v$.
- Case 3: $K \subseteq S_x$ and $|S_x| \leq v$.

In Case 1, each of the Q trials fail, and so D outputs a random bit, independent of i^*. Thus, such queries reveal no information about i^*.

In Case 2, for each of the Q trials of D, we can calculate the probability that trial will succeed. The trial chooses $t-a+1$ random elements of $[n] \setminus (K \cup \{i^*\})$, and succeeds if those elements all happen to be inside the set $S_x \setminus (K \cup \{i^*\})$, which has at least $v-a+1$ elements because we are in Case 2. Thus, for each trial the probability of success is $p = \binom{v-a+1}{t-a+1} / \binom{n-a}{t-a+1}$. Note that, by our definition of Q, we have $Q = \lambda/p$. Each of the λ/p trials are independent, and thus the probability

of failure over all trials is $(1-p)^{\lambda/p} \leq e^{-\lambda}$. Thus, union-bounding over the polynomially-many queries made by Trace, with overwhelming probability all Case 2 queries will succeed.

We let bad be the event that either (1) Trace makes a Case 2 query that fails, or (2) Trace makes a Case 3 query. We observe that if bad does not occur, then the view of Trace is perfectly independent of the choice of $i^* \in [n] \setminus K$.

But now we argue that, unless the index set $\mathcal{I}$ for sets S_x is large, for most K there will simply not exist a Case 3 query. In particular, each S_x can be a Type 3 query for at most $\binom{v}{a-1}$ different K. But there are $\binom{n}{a-1}$ different choices of K. Therefore, the probability that there is even a possible Case 3 query for K is $\leq |\mathcal{I}| \cdot \binom{v}{a-1} / \binom{n}{a-1}$. The probability of bad is at most negligibly-larger than this quantity, due to Case 2. Thus, we have that

$$\begin{aligned}
\delta(\lambda) &\geq \Pr[i = i^*] \\
&\geq \Pr[i \neq \bot \wedge \neg \mathsf{bad}] \Pr[i = i^* | i \neq \bot \wedge \neg \mathsf{bad}] \\
&\geq (1 - \Pr[i = \bot] - \Pr[\mathsf{bad}]) \Pr[i = i^* | i \neq \bot \wedge \neg \mathsf{bad}] \\
&\geq \left(1 - \delta(\lambda) - \mathsf{negl}(\lambda) - \frac{|\mathcal{I}| \binom{v}{a-1}}{\binom{n}{a-1}}\right) \frac{1}{n-a+1}
\end{aligned}$$

Rearranging gives

$$\begin{aligned}
|\mathcal{I}| &\geq (1 - \mathsf{negl}(\lambda) - \delta(\lambda)(n-a+2)) \times \frac{\binom{n}{a-1}}{\binom{v}{a-1}} = (1 - \mathsf{negl}(\lambda)) \times \frac{n(n-1)\cdots(n-a+2)}{v(v-1)\cdots(v-a+2)} \\
&\geq (1 - \mathsf{negl}(\lambda)) \left(\frac{n}{v}\right)^{a-1} \geq (1 - \mathsf{negl}(\lambda)) \times \left(\frac{Q}{e\lambda}\right)^{\frac{a-1}{t-a+1}}
\end{aligned}$$

This means

$$|\mathsf{ct}| \geq \log_2 |\mathcal{I}| \geq \frac{a-1}{t-a+1} \log(Q/e\lambda) + \log_2(1 - \mathsf{negl}(\lambda)) \geq \frac{a-1}{t-a+1} \log(Q/e\lambda) - 1 \tag{1}$$

In the case where $a = t - O(1)$, we can take $Q/\lambda \geq \Omega(\lambda)$, which gives a lower-bound $|\mathsf{ct}| \geq \Omega(a \log \lambda) = \Omega(t \log \lambda)$.

For $a = t - \omega(1)$, recall that once we fix the parameters of the traitor tracing scheme, we can take $Q/e\lambda$ to be an arbitrarily large polynomial. In particular, for any supposed upper bound on $|\mathsf{ct}|$ of the form $|\mathsf{ct}| \leq \frac{a-1}{t-a+1} \times O(\log \lambda)$, we can choose $Q/e\lambda$ large enough so that Eq 1 contradicts that bound (at least, assuming $a > 1$ so as to absorb the -1 in Eq 1). Thus, we must have $|\mathsf{ct}| \geq \frac{a-1}{t-a+1} \times \omega(\log \lambda)$. □

4.1 The Universal Tracing model

In this subsection, we consider TTT schemes with a universal tracing algorithm, i.e. the algorithm is independent of the behavior of the box. In particular, it does not depend on the runtime of the box and hence, we can trace boxes which run in super polynomial time. Theorem 9 below presents a stronger lower bound for TTT schemes with universal tracing.

Theorem 9. *Let TTT be a subset-based universal threshold traitor tracing scheme that is (a, ϵ, δ)-secure for any negligible δ, $a \leq t$ and any $\epsilon \geq 0$. Then, $|\mathsf{ct}| \geq (a-1)\log_2(n/t) - 1$.*

Proof (of Theorem 9). Consider the following adversary $\mathcal{A}$:

- First choose a random subset $K \subseteq [n]$ of size $a-1$.
- Choose a random $i^* \in [n] \setminus K$. $\mathcal{A}$ will query secret keys for all parties $i \neq i^*$.
- $\mathcal{A}$ will the choose two random messages μ_0, μ_1, and output the decoder D which behaves in the following way on input ct:
  - For each $i \in [n] \setminus \{i^*\}$, run $d_i \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$.
  - For *all* subsets $\mathcal{J} \subseteq [n] \setminus \{i^*\}$ of size t such that $K \subseteq \mathcal{J}$: run $\mu' \leftarrow \mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \mathcal{J}, \{d_j\}_{j \in \mathcal{J}})$, If $\mu' = \mu_b$ for some b, output b.
  - If D has not yet output anything, output a random bit.

Crucially, D tries to decrypt with *all* different subsets $\mathcal{J}$ that are larger than K (as opposed to only trying Q random subsets as in the proof of Theorem 8). If any of them successfully decrypt, D will output that result. Observe that this box may run in time super-polynomial in the security parameter, because it may try upto $\binom{n-a}{t-a+1}$ different subsets in the worst case.

As before, we will show that unless Trace can query D on ciphertexts corresponding to authorized user sets S_x that are "close" to K, Trace will be unable to learn any information about i^*, other than the fact that it is outside K.

Let i be the smallest user outside of K that is accused by Trace, if no such user exists, $i = \perp$. First, observe that $\Pr[\mathsf{GoodDec}] = 1$ and so $\Pr[\mathsf{GoodTr}] \geq 1 - \delta(\lambda)$. This implies $\Pr[i = \perp] \leq \delta(\lambda)$. Next, $\Pr[\mathsf{BadTr}] \leq \delta(\lambda)$ which means $\Pr[i = i^*] \leq \delta(\lambda)$.

We now claim that i^* is statistically hidden to Trace. Consider a query ct to D, associated with an authorized user set S_x. We break into three possibilities:

- Case 1: $K \not\subseteq S_x$.
- Case 2: $K \subseteq S_x$ but $|S_x| > t$.
- Case 3: $K \subseteq S_x$ and $|S_x| \leq t$.

As before, in Case 1, D outputs a random bit independent of i^*, so such queries reveal no information about i^*. In Case 2, S_x contains at least $t+1$ parties, and since only one party is honest, S_x contains at least t corrupt parties, hence, D will be able to successfully decrypt for any Case 2 query. Hence the tracer learns nothing from any Case 2 queries either.

Let bad be the event that Trace makes a Case 3 query. Based on the arguments above, if the event bad does not occur, then the view of the tracer is perfectly independent of the choice of i^*. But we show below that unless the index set $\mathcal{I}$ for sets S_x is large, for most K, there will simply not exist a Case 3 query. In particular, each S_x can be a Case 3 query for at most $\binom{t}{a-1}$ different subsets K, but there are $\binom{n}{a-1}$ different choices of K. Therefore, the probability that there exists a Case 3 query for K is

$$\leq \frac{|\mathcal{I}|\binom{t}{a-1}}{\binom{n}{a-1}}.$$

Thus we have that,

$$\begin{aligned}
\delta(\lambda) &\geq \Pr[i = i^*] \\
&\geq \Pr[i \neq \perp \wedge \neg \mathsf{bad}] \cdot \Pr[i = i^* | i \neq \perp \wedge \neq \mathsf{bad}] \\
&\geq \left(1 - \delta(\lambda) - \frac{|\mathcal{I}|\binom{t}{a-1}}{\binom{n}{a-1}}\right) \cdot \frac{1}{n-a+1}
\end{aligned}$$

Rearranging gives,

$$|\mathcal{I}| \geq \frac{\binom{n}{a-1}}{\binom{t}{a-1}} \cdot (1 - \delta(\lambda)(n-a+2))$$
$$\geq (1 - \mathsf{negl}(\lambda)) \left(\frac{n}{t}\right)^{a-1}.$$

This means,

$$|\mathsf{ct}| \geq \log_2 |\mathcal{I}| \geq (a-1)\log(n/t) + \log_2(1 - \mathsf{negl}(\lambda)) \geq (a-1)\log(n/t) - 1.$$

In particular, this implies that for $a = t - O(1)$, the ciphertext must scale proportionally to $\Omega(t)$. □

5 Lower-bound on Fingerprinting Code-based Traitor Tracing

In this section, we give a lower-bound on fingerprinting codes indicating that threshold traitor tracing schemes based on such codes are unlikely to be able to have ciphertext much smaller than $\Omega(n)$.

Recall that a (plain) binary fingerprinting code is the following:

Definition 4. *A binary fingerprinting code is a pair of* PPT *algorithms* $\mathsf{Code} = (\mathsf{FCGen}, \mathsf{FCTrace})$ *defined as follows:*

- $\mathsf{FCGen}(1^n, 1^{1/\epsilon}, 1^{1/\delta}) \to (\Gamma := \bar{w}^{(1)}, \ldots, \bar{w}^{(n)}, \mathsf{tk})$ *is the probabilistic code generator algorithm, that takes as input the number of parties, a security parameter ϵ and a noise bound parameter δ, and outputs n codewords $w_i \in \{0,1\}^\ell$ where $\ell = \ell(n, \epsilon, \delta)$ is a parameter, along with a tracing key* tk.
- $\mathsf{FCTrace}(\mathsf{tk}, w^*) \to \mathcal{S}$ *is the deterministic tracing algorithm, that takes as input a string $w^* \in \{0,1,?\}^\ell$ and the tracing key* tk*. It outputs a subset $\mathcal{S} \subseteq [n]$. Informally, elements in $\mathcal{S}$ are "accused" of creating the word w^*.*

Before defining security for a fingerprinting code, let us introduce some notation. A word $\bar{w} \in \{0,1\}^\ell$ is a binary string, and we use w_i to denote the ith letter of $\bar{w}$ for $i = 1, \ldots, \ell$; that is $\bar{w} = w_1 w_2 \ldots w_\ell$. A *noisy* word is a ternary string $\bar{w} \in \{0, 1, \text{'?'}\}^\ell$, and we similarly write $\bar{w} = w_1 w_2 \ldots w_\ell$ where each w_i is in $\{0, 1, \text{'?'}\}$.

Let $W = \{\bar{w}^{(1)}, \ldots, \bar{w}^{(f)}\}$ be a set of words in $\{0,1\}^\ell$. We say that a noisy word $\bar{w}^*$ is feasible for W if, for all $i \in [\ell]$, either $w_i = \text{'?'}$ or there is a $j \in [f]$ such that $w_i^* = w_i^{(j)}$. That is, if all the words in W have 0 as their ith bit, then $\bar{w}^*$ must have either 0 or '?' in its ith bit as well (the analogous condition should for 1). For example, if $W = \{00110, 10100\}$, then the words that are feasible for W are $00100, 00110, 10100, 10110$ and any word obtained from any of these four by replacing a subset of the bits with '?'. For a set W of words, the *feasible set* for W, denoted $F(W)$, is the set of all noisy words feasible for W. For $\delta \in [0,1]$, we say that a word is δ-noisy if at most a δ-fraction of its characters is '?', and for a set W of words we denote $F_\delta(W) = \{\bar{w} \in F(W) \ : \ \bar{w} \text{ is } \delta\text{-noisy}\}$.

Definition 5 (Security for a fingerprinting code). *A fingerprinting code* Code *is said to be* **fully collusion resistant** *if for all algorithms $\mathcal{A}$, all $n > 0$, all $\epsilon \in (0,1)$, all $\delta \in [0,1]$, and all subsets $\mathcal{I} \subseteq [n]$ it holds that*

$$\mathsf{Adv}^{\mathrm{cr}}_{\mathcal{A},\mathsf{Code}}(n, \mathcal{I}, \delta) := \Pr[\mathbf{ExpFC}_{\mathcal{A},\mathsf{Code}}(n, \mathcal{I}, \epsilon, \delta) = 1] < \epsilon.$$

Experiment $\mathbf{ExpFC}_{\mathcal{A},\mathsf{Code}}(n, \mathcal{I}, \epsilon, \delta)$

$(\Gamma, \mathsf{tk}) \leftarrow\!\!\$\ \mathsf{FCGen}(1^n, \epsilon, \delta)$ // write $\Gamma = \{\bar{w}^{(1)}, \ldots, \bar{w}^{(n)}\}$
$W \leftarrow \{\bar{w}^{(i)}\}_{i \in \mathcal{I}}$
$\bar{w}^* \leftarrow\!\!\$\ \mathcal{A}(W)$
if $\bar{w}^* \notin F_\delta(W)$ **then return** 0
$S \leftarrow \mathsf{FCTrace}(\mathsf{tk}, \bar{w}^*)$
if $S = \emptyset \ \vee\ S \not\subseteq \mathcal{I}$ **then return** 1 **else return** 0

Fig. 4. The security experiment for a fingerprinting code Code and an adversary $\mathcal{A}$. The integer n indicates the number of words in the code, the subset $\mathcal{I} \subseteq [n]$ the set of corrupted codewords, the parameter $\epsilon \in [0, 1]$ specifies the desired error bound on the tracing success, and $\delta \in [0, 1]$ the fraction of unknown ('?') locations.

We described the tracing mechanism for fingerprinting-code based traitor tracing (i.e. $t = 1$) in Section 2.2, but for the case of threshold traitor tracing, a more sophisticated cryptographic mechanism is needed to enforce the threshold requirements on top of the fingerprinting code structure. In any case, the ciphertext size of the resulting construction must scale with $|\Sigma|$, since there is a separate, independent ciphertext for each symbol. Thus, to get small ciphertexts, it is necessary to have small alphabets. Note that one can try to use more sophisticated mechanisms that do not have independent ciphertexts for each symbol, in hopes of getting more compact ciphertexts. For example, in plain traitor tracing, a PLBE system for each position could be used, instead of $|\Sigma|$ independent public keys. However, such mechanisms usually end up being themselves already solutions to the traitor tracing problem. For example, PLBE already gives traitor tracing directly. Hence, we will focus on lower-bounding the alphabet size $|\Sigma|$ to get a lower-bound on the ciphertext size.

If we wish to recover more than one user, we need a fingerprinting code capable of identifying multiple users. As described in Section 2.2, we consider a scenario wherein the cryptography allows us to extract a subset of symbols for each position (instead of just a single symbol), such that the box must possess at least t codewords whose symbols at that position belong in the subset. For instance, if we get a '1' at a position $j \in [\ell]$, then the box must have t parties whose codeword has a '1' at this position. We now define a generalization of fingerprinting codes wherein we can trace t parties given this guarantee.

Definition 6. *A multi-fingerprinting code with alphabet Σ is a pair of* PPT *algorithms* $\mathsf{mCode} = (\mathsf{mFCGen}, \mathsf{mFCTrace})$ *defined as follows:*

- $\mathsf{mFCGen}(1^n, 1^t, 1^a, 1^{1/\epsilon}, 1^{1/\delta}) \to (\bar{w}^{(1)}, \ldots, \bar{w}^{(n)}, \mathsf{tk})$ *is the probabilistic code generator algorithm, that takes as the number of parties, the threshold t, the minimal number of accused users a, a security parameter ϵ and a noise bound parameter δ, and outputs n codewords $w_i \in \Sigma^\ell$ where $\ell = \ell(n, t, a, \epsilon, \delta)$ is a parameter, along with a tracing key* tk.
- $\mathsf{mFCTrace}(\mathsf{tk}, w^*) \to \mathcal{S}$ *is the deterministic tracing algorithm, that takes as input a multi-string $w^* \in \{2^\Sigma\}^\ell$ and the tracing key* tk *(here 2^Σ denotes all subsets of Σ, and we will interpret $\emptyset$ as a '?'). It outputs a subset $\mathcal{S} \subseteq [n]$. Informally, elements in $\mathcal{S}$ are "accused" of creating the multi-string w^*.*

Informally, the security guarantee for multi-fingerprinting code requires that the $\mathsf{mFCTrace}$ algorithm must output at least a traitors, given that the multi-string w^* is multi-feasible for the

set of codewords of corrupt parties. More formally, let $W = \{\bar{w}^{(1)}, \ldots, \bar{w}^{(f)}\}$ be a set of words with $f \geq t$. We say that a noisy multi-word $\bar{w}^*$ is t-feasible for W if for all $i \in [\ell]$, either $w_i^* = \emptyset$ (i.e. '?') or there are at least t words $j_1 \neq j_2 \ldots \neq j_t \in [f]$ such that $w_i^{(j_k)} \in w_i^*$ for all $k \in [t]$. For a set of words W, we define $F_{\delta,t}(W)$ as the set of all noisy multi-words that are t-feasible for W, with at most a δ-fraction of '?'. We now formally define security of a multi-fingerprinting code.

Definition 7 (Security for a multi-fingerprinting code). *A multi-fingerprinting code* mCode *is said to be* **fully collusion resistant** *if for all algorithms* $\mathcal{A}$*, all* $0 < a \leq t \leq n$*, all* $\epsilon \in (0,1)$*, all* $\delta \in [0,1]$*, and all subsets* $\mathcal{I} \subseteq [n]$ *it holds that*

$$\mathsf{Adv}^{\mathrm{cr}}_{\mathcal{A},\mathsf{mCode}}(n,t,a,\mathcal{I},\delta) := \Pr[\mathbf{ExpmFC}_{\mathcal{A},\mathsf{mCode}}(n,t,a,\mathcal{I},\epsilon,\delta) = 1] < \epsilon.$$

Experiment $\mathbf{ExpmFC}_{\mathcal{A},\mathsf{mCode}}(n,t,a,\mathcal{I},\epsilon,\delta)$ // $|\mathcal{I}| \geq t$

1 : $(\Gamma, \mathsf{tk}) \leftarrow\!\$\ \mathsf{mFCGen}(1^n, 1^t, 1^a, \epsilon, \delta)$ // write $\Gamma = \{\bar{w}^{(1)}, \ldots, \bar{w}^{(n)}\}$
2 : $W \leftarrow \{\bar{w}^{(i)}\}_{i \in \mathcal{I}}$
3 : $\bar{w}^* \leftarrow\!\$\ \mathcal{A}(W)$
4 : **if** $\bar{w}^* \notin F_{\delta,t}(W)$ **then return** 0
5 : $S \leftarrow \mathsf{mFCTrace}(\mathsf{tk}, \bar{w}^*)$
6 : **if** $S \not\subseteq \mathcal{I} \vee |S| < a$ **then return** 1 **else return** 0

Fig. 5. The security experiment for a multi-fingerprinting code mCode and an adversary $\mathcal{A}$. The integer n indicates the number of words in the code, t indicates the threshold, the subset $\mathcal{I} \subseteq [n]$ denotes the set of corrupted codewords, the parameter $\epsilon \in [0,1]$ specifies the desired error bound on the tracing success, and $\delta \in [0,1]$ is the fraction of unknown ('?') locations.

We now prove that any multi-fingerprinting code allowing for tracing many users must have either a large alphabet or an *exponentially*-large length. Thus, without employing additional mechanisms, approaches based on multi-fingerprinting codes seem unlikely to yield schemes with short ciphertexts.

Theorem 10. *For polynomial* $t = \lambda^{\Omega(1)}$*, either* $|\Sigma| \geq \frac{1}{4} \times (n/t)^{a/t} \times t$ *or* $\ell \geq 2^{\Omega(t)}$.

Implications. In the case $a = t$, we must have $|\Sigma| \geq n/4$ unless the code-length is exponential. Note that $|\Sigma| = n$ is straightforward, as this case corresponds to the trivial threshold traitor tracing scheme [BPR24a]. Theorem 10 above shows that for $a = t$, the trivial case is asymptotically optimal.

Note that Theorem 10 gives a stronger lower-bound than Theorem 8, at the cost of only applying to schemes based on fingerprinting codes. In particular, observe that Theorem 8 allows for ciphertexts of size $O(t)$ (times factors polynomial in λ) even in the case $a = t$, whereas Theorem 10 shows that ciphertexts must have size $\Omega(n)$.

Even for $a < t$, Theorem 10 imposes a rather strong lower-bound. For example, when $a = t(1-\lambda^{-c})$, Theorem 8 allows for ciphertexts of size polynomial in λ, and hence ceases to give strong lower-bounds. However, Theorem 10 still results in $|\Sigma| \geq \Omega(n)$. For a/t even smaller than $1-\lambda^{-c}$, Theorem 8 continues to be un-interesting, while Theorem 10 continues to be non-trivial at long as $a = \Omega(t)$.

We note that it may be possible to partially circumvent our lower-bound using more advanced cryptosystems. In particular, it may be possible to consider an exponentially-large fingerprinting code, such that the codewords and decryption keys can be succinctly represented. However, it still seems necessary to send the position $j \in [\ell]$ as part of the ciphertext, which takes $\log_2 \ell$ bits to describe. We emphasize that we do not currently know how to achieve such a scheme, or whether such a scheme is possible. But even so, Theorem 10 still imposes significant constraints on such a hypothetical mechanism. For example, for $a/t = 1 - \lambda^{-c}$, Theorem 10 shows that either $|\Sigma| \geq \Omega(n)$ (giving $\Omega(n)$-sized ciphertext) or $\ell \geq 2^{\Omega(t)}$, giving $\Omega(t)$-sized ciphertexts.

We now give the proof of Theorem 10.

Proof. Consider the adversary with the words for all indexes but i^*. It will pick random indexes $i_1, \cdots, i_{a-1} \neq i^*$ to sacrifice. It creates the adversarial multi-codeword w^* as follows. Let S_j be the set of symbols at position j from among $\{w_j^{i_k}\}_{k \in [a-1]}$. If the adversary's codewords at position j contain at least t symbols from S_j (counting multiplicities), then the adversary outputs S_j at that position. Otherwise, it outputs an arbitrary set containing at least t of its symbols.

We will say a position $j \in [\ell]$ is good if, among all users (including i^*) there are at least $t+1$ codewords whose symbol at position j is in S_j. A bad position is one for which there are at most t codewords whose jth symbol is in S_j.

Notice that, for good j, the adversary outputs S_j, independent of the choice of i^*. Hence, only the symbol subsets at the bad positions can leak information about the honest index i^*.

We now claim that, unless the code is exponentially long or the symbol set Σ is large, the probability of any position being bad is small. Thus, no information about i^* is revealed to Trace.

Let us look at any particular position j, and analyze the probability that j is bad. For each symbol $\sigma \in \Sigma$, let T_σ^j be the set of indices i such that $w_j^i = \sigma$. Then $\{T_\sigma^j\}_{\sigma \in \Sigma}$ partitions the set of indices into $|\Sigma|$ parts.

Now, $i_1, \cdots, i_{a-1}$ are chosen at random, independent of of the partition T_σ^j. We will say that T_σ^j is hit if $i_k \in T_\sigma^j$ for some i_k. In order for j to be bad, the union of the hit parts must have size at most t. Now we will show that this is unlikely. This follows from the tension between two competing forces affecting the size of the union: if the hit parts are small, it is unlikely that all of the $a-1$ users will lie in their union, while if the hit parts are large, then the union will be large, making j good. We now make this precise by giving a crude upper-bound on the probability.

First, note that if there are any empty T_σ^j, we can define a new alphabet $\Sigma' \subseteq \Sigma$ containing only the σ for which T_σ^j is non-empty, and apply the following arguments to Σ', which will only improve our bounds. So we assume that each T_σ^j is non-empty.

Now, consider a subset $S \subseteq \Sigma$, and consider the union $\cup_{\sigma \in S} T_\sigma^j$. If $|\cup_{\sigma \in S} T_\sigma^j| > t$, then if $S_j = S$, j is automatically good.

Therefore, we consider S such that $|\cup_{\sigma \in S} T_\sigma^j| \leq t$, which in particular implies that $|S| \leq t$. For such S, $\Pr[S_j = S] \leq \Pr[S_j \subseteq S]$, which in turn is the probability that each of the $a-1$ random

choices of $i_1, \cdots, i_{a-1}$ lie in $\cup_{\sigma \in S} T_\sigma^j$. This in turn is equal to

$$\begin{aligned}
\Pr[S_j \subseteq S] &\leq \binom{|\cup_{\sigma \in S} T_\sigma^j|}{a-1} \Big/ \binom{n}{a-1} \\
&\leq \binom{t}{a-1} \Big/ \binom{n}{a-1} \\
&= \frac{t(t-1)\cdots(t-a+2)}{n(n-1)\cdots(n-a+2)} \leq \left(\frac{t}{n}\right)^{a-1}
\end{aligned}$$

Now, there are at most $\sum_{s=0}^{t} \binom{|\Sigma|}{s}$ such sets S. Union-bounding over all such sets gives an overall probability

$$\begin{aligned}
p := \Pr[|\cup_{\sigma \in S_j} T_\sigma^j| \leq t] &\leq \left[\sum_{s=0}^{t} \binom{|\Sigma|}{s}\right] \times \left(\frac{t}{n}\right)^{a-1} \\
&\leq \left(\frac{e|\Sigma|}{t}\right)^{t} \times \left(\frac{t}{n}\right)^{a-1}
\end{aligned}$$

The last inequality follows from the standard bound on the binomial coefficient $\sum_{s=0}^{t} \binom{S}{s} \leq (eS/t)^t$. For a proof of this inequality, see e.g. Proposition 1.4 in [Juk01].

Now, suppose $|\Sigma| \leq \frac{1}{4} \times (n/t)^{a/t} \times t$. Then we have

$$\begin{aligned}
p &\leq (e/4)^t \times \left(\frac{n}{t}\right)^{a} \times \left(\frac{t}{n}\right)^{a-1} \\
&\leq (e/4)^t \times (n/t) \\
&\leq 2^{-\Omega(t)}
\end{aligned}$$

Thus, in the event that $|\Sigma| \leq \frac{1}{4} \times (n/t)^{a/t} \times t$, if we union-bound over all ℓ indices, the probability that *some* index is bad is at most $\ell \times 2^{-\Omega(t)}$. Thus, unless $\ell \geq 2^{\Omega(t)}$, the probability is negligible, and tracing will fail. □

6 Upper bound for tracing many users

In this section, we present our construction that generalizes PLBE techniques to construct traitor tracing for general access structures, wherein we can trace an authorized set of corrupt parties. We start with formally defining this primitive below.

6.1 Traitor Tracing for general access structures

A traitor tracing scheme for general access structures is defined similar to a threshold traitor tracing scheme, but decryption requires a set of users authorized with respect to the access structure. More formally, a traitor tracing scheme for a circuit ensemble $\{\mathcal{C}_\lambda\}_\lambda$ is a tuple of five PPT algorithms $\mathsf{TTG} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine}, \mathsf{Trace})$. These algorithms are exactly as defined in Section 3 but with a slight change to the KeyGen algorithm:

– $\mathsf{KeyGen}(1^\lambda, C) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{tk})$ is the probabilistic key generation algorithm, that takes as input the security parameter, the access structure in the form of a polynomial-sized monotone circuit $C : \{0,1\}^n \to \{0,1\}$ such that $C \in \mathcal{C}_\lambda$. It outputs a public key pk, secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ along with the tracing key tk.

We do not have a as an additional input in KeyGen – we will require that the tracing algorithm finds an authorized set of users. Note that we focus on *monotone* access structures, wherein if a set $S \subseteq [n]$ is authorized with respect to C, then all supersets of S are also authorized.

Correctness. An honestly generated ciphertext should be correctly decrypted by any authorized subset of decryptors. We formalize this notion in Definition 8 below.

Definition 8. *A traitor tracing scheme* TTG *for a circuit family* $\{\mathcal{C}_\lambda\}_\lambda$ *is said to be correct if there exists a negligible function* $\mathsf{negl}(\lambda)$ *of the security parameter* λ*, such that, for all* $\lambda \in \mathbb{N}$*, all* $C \in \mathcal{C}_\lambda$*, all sized subsets* $\mathcal{J} \subseteq [n]$ *such that* $C(x_\mathcal{J}) = 1$*, and all messages* μ *in the message space, it holds that*

$$\Pr\left[\mu' = \mu \;:\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{tk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C) \\ c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mu) \\ \forall\ i \in \mathcal{J},\ d_i \leftarrow \mathsf{Dec}(\mathsf{sk}_i, c) \\ \mu' \leftarrow \mathsf{Combine}(\mathsf{pk}, c, \{d_j\}_{j \in \mathcal{J}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Semantic Security. We define semantic security in the same way as in Figure 3, with the only change being that the adversary outputs the access structure $C \in \mathcal{C}_\lambda$ instead of (n, t), and it is allowed to corrupt any unauthorized subset of parties. For completeness, we present the formal definition below:

Definition 9 (CPA security). *We say that* TTG *satisfies semi-adaptive CPA security if for every probabilistic polynomial time adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{sa-cpa}}_{\mathcal{A},\mathsf{TTG}}(\lambda) := \left|\frac{1}{2} - \Pr[\mathbf{IND\text{-}CPA}^{sa}_{\mathcal{A},\mathsf{TTG}}(\lambda) = 1]\right|.$$

Traceability. Similar to Section 3, if the decoder D, given a ciphertext ct, can learn any information about the encrypted message, then Trace must trace D to an authorized subset of the coalition that "manufactured" D. Moreover, it should not falsely accuse any honest user.

Definition 10. *Let* $\epsilon = \epsilon(\lambda)$ *and* $\delta = \delta(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$*. A traitor-tracing scheme for general access structures* $\mathsf{TTG} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine}, \mathsf{Trace})$ *with key space* $\mathcal{K} = \{\mathcal{K}_\lambda\}$ *is* (ϵ, δ)*-secure if for every* PPT *adversary* $\mathcal{A}$ *and for every* $\lambda \in \mathbb{N}$*, the following holds. Consider the following experiment:*

– $(C, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$
– $(\mathsf{pk}, \mathsf{sk}_1, \cdots, \mathsf{sk}_n, \mathsf{tk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C)$
– $(D, \mu_0, \mu_1) \leftarrow\!\$\ \mathcal{A}^{\mathsf{sk}(\cdot)}(\mathsf{state}, \mathsf{pk})$*, where* $\mathsf{sk}(i)$ *returns* sk_i*. Let* $\mathcal{J}$ *be the set of queries* $\mathsf{sk}(i)$ *made by* $\mathcal{A}$*.*
– *Let* $\mathcal{J}' \leftarrow\!\$\ \mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^{1/\epsilon(\lambda)}, \mu_0, \mu_1)$*.*
– *Define:*
  - $\mathsf{GoodDec}$ *is the event that* $\Pr[D(c) = b : b \leftarrow\!\$\ \{0,1\}, c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mu_b)] \geq 1/2 + \epsilon(\lambda)$
  - GoodTr *is the event that* $C(x_{\mathcal{J}'}) = 1$*.*

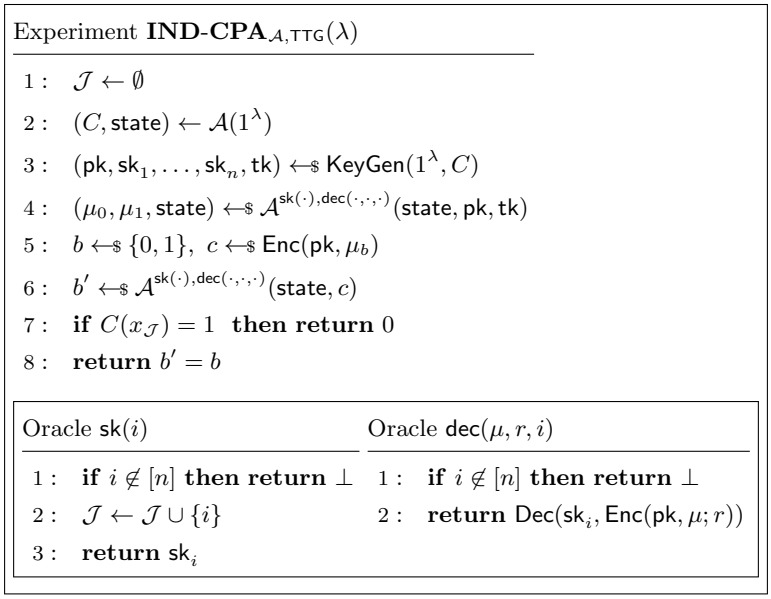


Fig. 6. The semantic security experiment for a traitor tracing scheme TTG for general access structures and an adversary $\mathcal{A}$.

- BadTr *is the event that* $\mathcal{J}' \not\subseteq \mathcal{J}$.

– *Then*

$$\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \delta(\lambda) \qquad \textit{and} \qquad \Pr[\mathsf{BadTr}] \leq \delta(\lambda)$$

Extensions. We define the following variations of the TTG definition.

Risky Traitor Tracing for General Access Structures. We define a risky traitor tracing scheme for general access structures similar to [GKRW18]; traitors are only accused with some small but noticeable probability.

Definition 11 (Risky Tracing). *Let $\alpha = \alpha(n, \lambda)$ be a polynomial. A traitor tracing scheme for General Access Structures* TTG *is α-risky if, for all* PPT *adversaries $\mathcal{A}$ and all inverse-polynomials ϵ, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda)$ *and* $\Pr[\mathsf{GoodTr}] \geq \alpha \cdot \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda)$.

Limited Traitor Tracing for General Access Structures. A Limited traitor tracing scheme [NP98] is one where malicious users are accused only for very good decoders that succeed a constant fraction of the time. We generalize this notion for traitor tracing for general access structures below. As mentioned in Section 2.3, this is traditionally referred to as 'threshold' traitor tracing, but we use Limited following [DDP+25].

Definition 12 (Limited Tracing). *A traitor tracing scheme for general access structures* TTG *is* Limited secure *if there exists a constant $\epsilon \in (0, 1/2)$ such that, for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda)$ *and* $\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda)$.

In the case of Limited secure schemes, the constant ϵ is hard-coded into the algorithm Trace, and we omit ϵ as an input to Trace. A TTT scheme that is δ-risky and ϵ-Limited may also be referred to as a (ϵ, δ)-risky Limited TTT scheme.

6.2 Private Set Broadcast Encryption for General Access Structures

A Private Set Broadcast Encryption for a circuit ensemble $\{\mathcal{C}_\lambda\}_\lambda$ is a tuple of five PPT algorithms $\mathsf{PSBEG} = (\mathsf{KeyGen}, \mathsf{EncPK}, \mathsf{EncSK}, \mathsf{Dec}, \mathsf{Combine})$:

- $\mathsf{KeyGen}(1^\lambda, C) \to (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]})$ is the probabilistic **setup** algorithm that takes as inputs the security parameter λ, the access structure in the form of a circuit $C \in \mathcal{C}_\lambda$, and outputs the public key pk, a master secret key msk and n secret keys $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n$. Note we assume that the circuit C implicitly defines the number of parties n.
- $\mathsf{EncPK}(\mathsf{pk}, \mu) \to \mathsf{ct}$ is the probabilistic **public key encryption** algorithm that takes as input the public key pk, a message μ and outputs a ciphertext ct.
- $\mathsf{EncSK}(\mathsf{msk}, \mu, j, \mathcal{T}) \to \mathsf{ct}$ is the **secret key encryption** algorithm that takes as inputs the master secret key msk, a message μ, an index $j \in [n]$ and a subset $\mathcal{T} \subseteq [n]$ and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}) \to d_i$ is the deterministic **decryption** algorithm that takes as input a secret key sk_i, a ciphertext ct and outputs the partial decryption d_i of ct under sk_i.
- $\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in\mathcal{J}}) \to \mu/\perp$ is the deterministic **combiner** algorithm that takes as input the public key pk, a ciphertext ct, a set $\mathcal{J} \subseteq [n]$ and the corresponding decryption shares $\{d_j\}_{j\in\mathcal{J}}$ and outputs either a message μ or $\perp$.

We remark that the notion of recipient-private broadcast encryption defined in [BZ14] can be seen as a special case of our definition, where any party can decrypt on its own.

Correctness. We require that an honestly generated ciphertext using the public key must be decrypted by any authorized subsets of parties. Moreover, an honestly generated secret-key ciphertext with respect to index j and set $\mathcal{T}$ must be decrypted by any authorized subset of $\mathcal{T} \cup \{i+1, \ldots, n\}$.

Definition 13 (Correctness). *A* PSBEG *scheme is said to be correct if there exists negligible functions* $\mathsf{negl}_1(\lambda)$, $\mathsf{negl}_2(\lambda)$ *such that, for all* $\lambda \in \mathbb{N}$, *all circuits* $C \in \mathcal{C}_\lambda$, *and for all subsets* $\mathcal{S} \subseteq [n]$ *such that* $C(x_\mathcal{S}) = 1$ *and for all messages* μ, *we have:*

$$\Pr\left[\mu' = \mu \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C) \\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncPK}(\mathsf{pk}, \mu) \\ d_i := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})\ \forall\ i \in \mathcal{S} \\ \mu' := \mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \mathcal{S}, \{d_j\}_{j\in\mathcal{S}})\end{array}\right] \geq 1 - \mathsf{negl}_1(\lambda)$$

Moreover, for all indices $j \in [n]$ *and all subsets* $\mathcal{T} \subseteq [n]$ *such that either* $j < i$ *or* $i \in \mathcal{T}$ *for all* $i \in \mathcal{S}$, *we have,*

$$\Pr\left[\mu' = \mu \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C) \\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu, j, \mathcal{T}) \\ d_i := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})\ \forall\ i \in \mathcal{S} \\ \mu' := \mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \mathcal{S}, \{d_i\}_{i\in\mathcal{S}})\end{array}\right] \geq 1 - \mathsf{negl}_2(\lambda).$$

Semantic Security. We generalize the semantic security definition of threshold PLBE as defined in [DDP+25] to the setting of general access structures. In particular, the adversary is allowed to corrupt any unauthorized subset of parties.

Definition 14 (CPA Security). *A* PSBEG *scheme is said to satisfy semi-adaptive strong CPA security if for every* PPT *adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{sa-cpa}}_{\mathcal{A},\mathsf{PSBEG}}(\lambda) := \left|\frac{1}{2} - \Pr[\mathbf{IND\text{-}CPA}^{sa}_{\mathcal{A},\mathsf{PSBEG}}(\lambda) = 1]\right|.$$

Experiment $\mathbf{IND\text{-}CPA}_{\mathcal{A},\mathsf{PSBEG}}(\lambda)$

$\mathcal{J} \leftarrow \emptyset$
$(C, \mathsf{state}) \leftarrow\$ \mathcal{A}(1^\lambda)$
$(\mathsf{pk}, \mathsf{msk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow\$ \mathsf{KeyGen}(1^\lambda, C)$
$(\mu_0, \mu_1, \mathsf{state}) \leftarrow\$ \mathcal{A}^{\mathsf{sk}(\cdot),\mathsf{dec}(\cdot,\cdot,\cdot)}(\mathsf{state}, \mathsf{pk}, \mathsf{msk})$
$b \leftarrow\$ \{0,1\},\ \mathsf{ct} \leftarrow\$ \mathsf{EncPK}(\mathsf{pk}, \mu_b)$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{sk}(\cdot),\mathsf{dec}(\cdot,\cdot,\cdot)}(\mathsf{state}, \mathsf{ct})$
if $C(x_{\mathcal{J}}) = 1$ **then return** 0
return $b' = b$

Oracle $\mathsf{sk}(i)$

if $i \notin [n]$ **then return** $\perp$
$\mathcal{J} \leftarrow \mathcal{J} \cup \{i\}$
return sk_i

Oracle $\mathsf{dec}(\mu, r, i)$

if $i \notin [n]$ **then return** $\perp$
return $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{EncPK}(\mathsf{pk}, \mu; r))$

Fig. 7. The semantic security experiment for a PSBE scheme PSBEG for general access structures and an adversary $\mathcal{A}$.

q-Bounded Security. The security of PSBEG is captured through three key properties. The first one is *indistinguishability*, which ensures that any PPT adversary controlling an arbitrary number of parties cannot distinguish if a ciphertext ct was generated using the public key encryption mode EncPK or using the secret key encryption mode EncSK with index $j = 0$ and set $\mathcal{T} = \emptyset$.

Second, we have *message hiding*, which requires that message privacy is preserved against an adversary controlling any number of parties for a ciphertext that has been encrypted using the secret key mode with index $j = n$ and any $\mathcal{T} \subseteq [n]$ as long as $C(x_{\mathcal{T}}) = 0$. Note that this notion of message hiding is stronger than that considered in previous works. For instance, for the threshold access structure, we require message privacy to hold even if up to $t - 1$ parties are authorized to decrypt the ciphertext, whereas prior definitions of threshold PLBE [DDP+25] only require message privacy with respect to a ciphertext for which no parties are authorized to decrypt. Additionally, the adversary is allowed to make upto q queries to the EncSK oracle, to get secret-key encryptions for arbitrary values of $(i, \mathcal{T})$. We also consider the adaptive version of this definition, wherein the adversary may query secret keys arbitrarily before choosing the set $\mathcal{T}$ for the challenge ciphertext.

The third property is called *set hiding*, which is a generalization of index hiding as defined for threshold PLBE [DDP+25] across two dimensions: (i) to the setting of general access structures and (ii) encrypting to an index $j \in [n]$ as well as a set $\mathcal{T}$. Herein, we require that the adversary cannot distinguish between an encryption to $(i_0, \mathcal{T}_0)$ and $(i_1, \mathcal{T}_1)$ as long as it only gets secret

keys for parties that are authorized to decrypt either both ciphertexts or none of the ciphertexts. More formally, we allow the adversary to query secret keys for any party $i \in [n] \setminus \mathcal{S}_0 \Delta \mathcal{S}_1$, where $\mathcal{S}_b = \{i_b+1, \ldots, n\} \cup \mathcal{J}_b$ is the set of parties authorized to decrypt ciphertexts encrypted to $(i_b, \mathcal{J}_b)$. Additionally, the adversary can make upto q queries to the EncSK oracle. Again, we consider adaptive version of this property.

Definition 15 (q-bounded Indistinguishability). *A PSBE scheme for general access structures* PSBEG *is said to satisfy q-bounded Indistinguishability if for any* PPT *adversary $\mathcal{A}$ that makes at most q many encryption queries to* $\mathsf{EncSK}(\mathsf{msk}, \cdot, 0, \emptyset)$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[b' = b \ : \ \begin{array}{c} (C, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}(1^\lambda) \\ (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C) \\ (\mu, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}^{\mathsf{EncSK}(\mathsf{msk},\cdot,0,\emptyset)}(\mathsf{state}, \mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}) \\ b \leftarrow\!\$\ \{0,1\} \\ \mathsf{ct}_0 \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu, 0, \emptyset), \mathsf{ct}_1 \leftarrow\!\$\ \mathsf{EncPK}(\mathsf{pk}, \mu) \\ b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{EncSK}(\mathsf{msk},\cdot,0,\emptyset)}(\mathsf{state}, \mathsf{ct}_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Note that here $\mathcal{A}$ is only allowed to query for secret-key ciphertexts corresponding to $(0, \emptyset)$.

Definition 16 (Adaptive q-bounded Message Hiding). *A PSBE scheme for general access structures* PSBEG *is said to satisfy adaptive q-bounded message hiding security if for any* PPT *adversary $\mathcal{A}$ that makes q many encryption queries to* $\mathsf{EncSK}(\mathsf{msk}, \cdot, \cdot, \cdot)$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that the following probability is bounded by* $1/2 + \mathsf{negl}(\lambda)$*:*

$$\Pr\left[C(x_{\mathcal{J}}) = 0 \wedge b' = b \ : \ \begin{array}{c} (C, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}(1^\lambda) \\ (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C) \\ (\mathsf{state}, \mu_0, \mu_1, \mathcal{J}) \leftarrow\!\$\ \mathcal{A}^{\mathsf{EncSK}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{state}, \mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}) \\ b \leftarrow\!\$\ \{0,1\},\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu_b, n, \mathcal{J}) \\ b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{EncSK}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{state}, \mathsf{ct}) \end{array}\right]$$

Definition 17 (Adaptive q-bounded Set Hiding). *A PSBE scheme for general access structures* PSBEG *is said to satisfy adaptive q-bounded set hiding security if for any* PPT *adversary $\mathcal{A}$ that makes q many encryption queries to* $\mathsf{EncSK}(\mathsf{msk}, \cdot, \cdot, \cdot)$ *and arbitrarily many* $\mathsf{skO}(i)$ *queries (which outputs the secret key of party i), there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that the following holds:*

$$\Pr\left[\mathbf{G}^{adap-set-hiding}_{\mathcal{A},\mathsf{PSBEG}}(\lambda) = 1\right] \leq 1/2 + \mathsf{negl}(\lambda)$$

Decoder-based Security. As observed in [GKW18], we need to consider decoder-based security definitions for constructing traitor tracing from PSBE. We start with defining the notion of *distinguishing decoders* for PSBEG with respect to different encryption modes.

Definition 18 (PSBEG Distinguisher). *For any $\gamma \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, a* PPT *decoder D, for* $\mathsf{params} \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C)$ *and for any message μ is said to be*

γ-$\mathsf{Dist}^{\mathsf{pk},(\mathsf{sk},0)}_{\mathsf{params}}$ *if*

$$\Pr\left[b' = b \ : \ \begin{array}{c} b \leftarrow\!\$\ \{0,1\} \\ \mathsf{ct}_0 \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu, 0, \emptyset), \mathsf{ct}_1 \leftarrow\!\$\ \mathsf{EncPK}(\mathsf{pk}, \mu) \\ b' \leftarrow\!\$\ D(\mathsf{ct}_b) \end{array}\right] \geq \frac{1}{2} + \gamma$$

Experiment $\mathbf{G}^{adap-set-hiding}_{\mathcal{A},\mathsf{PSBEG}}$

$(C, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}(1^\lambda)$

$(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, C)$

$\mathcal{Q} \leftarrow \emptyset\ ;\ \mathsf{count} = 0$

$(\mu, i_0, \mathcal{T}_0, i_1, \mathcal{T}_1, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}^{\mathsf{EncSKO}(\cdot,\cdot,\cdot),\mathsf{skO}(\cdot)}(\mathsf{state}, \mathsf{pk})$

$b \leftarrow\!\$\ \{0,1\},\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu, i_b, \mathcal{T}_b)$

$b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{EncSKO}(\cdot,\cdot,\cdot),\mathsf{skO}(\cdot)}(\mathsf{state}, \mathsf{ct})$

$\mathcal{S}_b := \{i_b + 1, \ldots, n\} \cup \mathcal{T}_b\ \forall\ b \in \{0,1\}$

return $b' = b \wedge \mathcal{Q} \cap (\mathcal{S}_0 \Delta \mathcal{S}_1) = \emptyset \wedge \mathsf{count} \leq q$

Oracle $\mathsf{EncSKO}(\mu, i, \mathcal{J})$:

$\mathsf{count} = \mathsf{count} + 1$

return $\mathsf{EncSK}(\mathsf{msk}, \mu, i, \mathcal{J})$

Oracle $\mathsf{skO}(i)$

$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{i\}$

return sk_i

Fig. 8. The adaptive q-query set hiding game for a PSBE scheme PSBEG.

γ-$\mathsf{Dist}^{i_0,\mathcal{J}_0,i_1,\mathcal{J}_1}_{\mathsf{params}}$ *if for indices* $i_0, i_1 \in [n]$ *and subsets* $\mathcal{J}_0, \mathcal{J}_1 \subseteq [n]$,

$$\Pr\left[b' = b\ :\ \begin{array}{c} b \leftarrow\!\$\ \{0,1\},\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu, i_b, \mathcal{J}_b) \\ b' \leftarrow\!\$\ D(\mathsf{ct}) \end{array}\right] \geq \frac{1}{2} + \gamma$$

γ-$\mathsf{Dist}^{n,\mathcal{J}}_{\mathsf{params}}$ *for any messages* μ_0, μ_1 *and any subset* $\mathcal{J} \subseteq [n]$ *such that* $C(x_{\mathcal{J}}) = 0$ *if,*

$$\Pr\left[b' = b\ :\ \begin{array}{c} b \leftarrow\!\$\ \{0,1\},\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu_b, n, \mathcal{J}) \\ b' \leftarrow\!\$\ D(\mathsf{ct}) \end{array}\right] \geq \frac{1}{2} + \gamma$$

We now define the decoder-based versions of all three security properties. We have decoder-based Indistinguishability, which says that no PPT adversary can produce a γ-$\mathsf{Dist}^{\mathsf{pk},(\mathsf{sk},0)}_{\mathsf{params}}$ decoder, even when given the secret keys of all n parties. While one could also define a stronger version of indistinguishability, the weaker notion considered here turns out to be sufficient for our constructions, as mentioned in Section 2.3.

We then define adaptive q-query decoder-based set hiding, which says that no adversary can produce a decoder that can distinguish between encryptions to $(i_0, \mathcal{J}_0)$, $(i_1, \mathcal{J}_1)$, even when given keys for all parties outside $\mathcal{S}_0 \Delta \mathcal{S}_1$ (as defined above). The adversary is allowed to additionally make q EncSK queries, and can choose the challenge index-set pairs adaptively. Lastly, we define adaptive q-query decoder-based message-hiding, wherein the adversary cannot produce a decoder that breaks semantic security for a ciphertext encrypted to index n and any unauthorized set $\mathcal{T}$ of its choice. As before, the adversary is allowed to additionally make q EncSK queries, and can choose the challenge set $\mathcal{T}$ adaptively.

Definition 19 (Decoder-based Indistinguishability). *A* PSBEG *scheme is said to satisfy decoder-based indistinguishability if for any* PPT *adversary* $\mathcal{A}$ *and for any non-negligible function* γ*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that the following probability is bounded by* $\mathsf{negl}(\lambda)$*:*

$$\Pr\left[D \text{ is } \gamma\text{-}\mathsf{Dist}^{\mathsf{pk},(\mathsf{sk},0)}_{\mathsf{params}} \text{ for } \mu\ :\ \begin{array}{c} (C, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}(1^\lambda) \\ (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, C) \\ (D, \mu) \leftarrow\!\$\ \mathcal{A}(\mathsf{state}, \mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}) \end{array}\right]$$

Definition 20 (Adaptive q-query Decoder-based Set Hiding Security). *A* PSBEG *scheme is said to satisfy adaptive decoder-based set hiding security if for any* PPT *adversary* $\mathcal{A}$ *and for any non-negligible function* γ*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that the following probability is bounded by* $\mathsf{negl}(\lambda)$*:*

$$\Pr\left[D \text{ is } \gamma\text{-}\mathsf{Dist}^{i_0,\mathcal{J}_0,i_1,\mathcal{J}_1}_{\mathsf{params}} \text{ for } \mu \;:\; \begin{array}{c}(C,\mathsf{state}) \leftarrow\!\$\; \mathcal{A}(1^\lambda)\\ (\mathsf{pk},\mathsf{msk},\{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\; \mathsf{Setup}(1^\lambda, C)\\ (D,\mu,i_0,\mathcal{J}_0,i_1,\mathcal{J}_1,\mathsf{state}) \leftarrow\!\$\; \mathcal{A}^{\mathsf{EncSKO}(\cdot,\cdot,\cdot),\mathsf{skO}(\cdot)}(\mathsf{state},\mathsf{pk})\end{array}\right],$$

where (i) the $\mathsf{skO}(i)$ *outputs* sk_i*, and the adversary is only allowed to query this oracle for* $i \notin \mathcal{S}_0 \Delta \mathcal{S}_1$*, where* $\mathcal{S}_b = \{i_b+1,\ldots,n\} \cup \mathcal{J}_b$ *for* $b \in \{0,1\}$*, (ii) the adversary can only make up to* q *queries to the* EncSKO *oracle.*

Definition 21 (Adaptive q-query Decoder-based Message Hiding Security). *A* PSBEG *scheme is said to satisfy decoder-based message hiding security if for any* PPT *adversary* $\mathcal{A}$ *and for any non-negligible function* γ*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that the following probability is bounded by* $\mathsf{negl}(\lambda)$*:*

$$\Pr\left[\begin{array}{c}C(x_{\mathcal{J}})=0 \wedge\\ D \text{ is } \gamma\text{-}\mathsf{Dist}^{n,\mathcal{J}}_{\mathsf{params}} \text{ for } \mu_0,\mu_1\end{array} \;:\; \begin{array}{c}(C,\mathsf{state}) \leftarrow\!\$\; \mathcal{A}(1^\lambda)\\ (\mathsf{pk},\mathsf{msk},\{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\$\; \mathsf{KeyGen}(1^\lambda, C)\\ (D,\mu_0,\mu_1,\mathcal{J}) \leftarrow\!\$\; \mathcal{A}^{\mathsf{EncSKO}(\cdot,\cdot,\cdot)}(\mathsf{state},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]})\end{array}\right],$$

where the adversary can only make up to q *queries to the* EncSKO *oracle.*

Remark 1 (On Selective Security). We also consider selective notions of security wherein the adversary has to specify the challenge set or index in the beginning of the game. In particular, for the message-hiding games, the adversary must output C as well as $\mathcal{J}$ in the beginning of the game. For the set-hiding games, the adversary must output $(i_0,\mathcal{T}_0),(i_1,\mathcal{T}_1)$ in the beginning of the game, along with C. Additionally, for the set-hiding definitions, we restrict the adversary to make all the EncSK oracle queries before querying any secret keys.

6.3 Traitor Tracing for General Access Structures from Adaptive poly(λ)-query Decoder-based PSBEG

In Figure 9, we present our generic compiler to construct TTG from a PSBE scheme for the same access structure, that achieves adaptive decoder-based security with $\mathsf{poly}(\lambda)$-queries.

Correctness. Correctness follows directly from correctness of PSBEG.

Semantic Security. Theorem 11 below proves semantic security of TTG by relying on semantic security of PSBEG.

Theorem 11. *For every* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda) = \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A},\mathsf{TTG}}(\lambda).$$

Proof (of Theorem 11). Suppose on the contrary, there exists an PPT adversary $\mathcal{A}$, non-negligible function $\gamma(\cdot)$, such that $\mathcal{A}$ breaks the strong IND-CPA security of TTG. Then, we can use $\mathcal{A}$ to build an adversary $\mathcal{B}$, that will use $\mathcal{A}$ to break the strong IND-CPA of the PSBE scheme PSBEG. $\mathcal{B}$ works as follows.

$\underline{\mathsf{KeyGen}(1^\lambda, C) \to (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) :}$

1. Let $(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\!\$\ \mathsf{PSBEG.KeyGen}(1^\lambda, C)$.
2. **output** $(\mathsf{pk}, \mathsf{tk} := \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]})$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu) \to \mathsf{ct} :}$

1. **output** $\mathsf{PSBEG.EncPK}(\mathsf{pk}, \mu)$.

$\underline{\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}) \to d_i :}$

1. **output** $\mathsf{PSBEG.Dec}(\mathsf{sk}_i, \mathsf{ct})$.

$\underline{\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in\mathcal{J}}) \to \mu :}$

1. **output** $\mathsf{PSBEG.Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in\mathcal{J}})$.

$\underline{\mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^y, \mu_0, \mu_1) :}$

1. Let $\epsilon := 1/y$ and $W := \lambda \cdot (n \cdot y)^2$.
2. Let $\mathcal{J} := \emptyset$ and $j = 1$. While $C(x_{\mathcal{J}}) = 0$ and $j < n$:
   (a) For $i = 0 \ldots n$
      i. Let $\mathsf{count}_{i,j} := 0$.
      ii. For $\ell = 1$ to W:
         A. Sample $b_{i,j,\ell} \leftarrow\!\!\$\ \{0, 1\}$.
         B. If $i = 0$, let $\mathsf{ct}_{i,j,\ell} \leftarrow\!\!\$\ \mathsf{PSBEG.EncSK}(\mathsf{msk}, \mu_{b_{i,j,\ell}}, 0, \emptyset)$.
         C. Else if $i > 0$, then $\mathsf{ct}_{i,j,\ell} \leftarrow\!\!\$\ \mathsf{PSBEG.EncSK}(\mathsf{msk}, \mu_{b_{i,j,\ell}}, i, \mathcal{J})$.
         D. If $D(\mathsf{ct}_{i,j,\ell}) = b_{i,j,\ell}$, update $\mathsf{count}_{i,j} := \mathsf{count}_{i,j} + 1$.
      iii. Let $\hat{p}_{i,j} := \mathsf{count}_{i,j}/W$.
   (b) Let $\mathcal{J}'_j := \{i \mid \hat{p}_{i-1,j} - \hat{p}_{i,j} \geq \epsilon/4n\}_{i\in[n]}$. Set $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}'_j$ and $j \leftarrow j + 1$.
3. **output** $\mathcal{J}$.

Fig. 9. Our compiler to construct a traitor tracing scheme for general access structures from PSBEG for the same access structure.

- $\mathcal{B}$ forwards 1^λ to $\mathcal{A}$ and receives (C, state). $\mathcal{B}$ then forwards them to its challenger.
- $\mathcal{B}$ receives $\mathsf{state}, \mathsf{pk}, \mathsf{msk}$ from its challenger, and forwards them to $\mathcal{A}$ as $(\mathsf{state}, \mathsf{pk}, \mathsf{tk}) := (\mathsf{state}, \mathsf{pk}, \mathsf{msk})$.
- $\mathcal{B}$ forwards all the secret key and partial decryption queries from $\mathcal{A}$ to its own oracle.
- Similarly, $\mathcal{B}$ forwards the challenge messages output by $\mathcal{A}$ to its challenger, and forwards the received response ct it receives from its challenger to $\mathcal{A}$.
- Finally, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs b' as its guess.

It is easy to see that $\mathcal{B}$ perfectly simulates the view of the $\mathcal{A}$. Therefore, if $\mathcal{A}$ breaks the IND-CPA security of TTG as per Definition 2 with some non-negligible probability $\gamma(\lambda)$, $\mathcal{B}$ breaks the strong IND-CPA security of PSBEG with probability $\gamma(\lambda)$. □

Traceability. We prove traceability of the compiler based on the indistinguishability and adaptive message-hiding and set-hiding properties of PSBEG.

Theorem 12. *Assuming* PSBEG *is a secure PSBE scheme for general access structures satisfying indistinguishability, adaptive* $\mathsf{poly}(\lambda)$*-query decoder-based set hiding, adaptive* $\mathsf{poly}(\lambda)$*-query decoder-based message-hiding, then the compiled scheme* TTG *is traceable as per Definition 10.*

Proof (of Theorem 12). We start with introducing some notation. Given any pirate decoder box D, let $\mathcal{J}_j$ denote the value of the set $\mathcal{J}$ at the beginning of the jth iteration of the tracing algorithm,

i.e. $\mathcal{J}_1 = \emptyset$. By construction, $\mathcal{J}_1 \subseteq \mathcal{J}_2 \subseteq \mathcal{J}_3$ and so on. Additionally, for messages μ_0, μ_1, for any iteration $j \in [n]$ and for any $i \in \{1, \ldots, n\}$, let

$$p^D_{i,j} = \Pr\left[D(\mathsf{ct}) = b : b \leftarrow\$ \{0,1\}, \mathsf{ct} \leftarrow\$ \mathsf{EncSK}(\mathsf{msk}, \mu_b, i, \mathcal{J}_j)\right],$$

where the probability is taken over random coins of decoder D as well as the randomness used during encryption. For $i = 0$, we simply replace $\mathcal{J}_j$ by $\emptyset$:

$$p^D_{0,j} = \Pr\left[D(\mathsf{ct}) = b : b \leftarrow\$ \{0,1\}, \mathsf{ct} \leftarrow\$ \mathsf{EncSK}(\mathsf{msk}, \mu_b, 0, \emptyset)\right].$$

Similarly, let $p^D_{nrml} = \Pr[D(\mathsf{ct}) = b : b \leftarrow\$ \{0,1\}, \mathsf{ct} \leftarrow\$ \mathsf{EncPK}(\mathsf{pk}, \mu_b)]$.

False Trace Probability. First, we show that the tracing algorithm never falsely accuses any honest user. Formally, we prove the following.

Theorem 13. *For every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda)$$

Proof. Let J be the set of keys queried by $\mathcal{A}$. For $j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, n\}$, we define events $\mathsf{DiffAdv}^D_{i,j} : p^D_{i-1,j} - p^D_{i,j} > \epsilon/8n$ and $\mathsf{DiffAdv}^D : \bigvee_{i \in [n] \setminus J, j \in [n]} \mathsf{DiffAdv}^D_{i,j}$. Lastly, let $\mathsf{DiffAdv}^D_i : \bigvee_{j \in [n]} \mathsf{DiffAdv}^D_{i,j}$.

As observed in [GKW18], the probability of the event false trace can be rewritten as follows by conditioning on the events defined above:

$$\Pr[\mathsf{BadTr}] \leq \Pr[\mathsf{BadTr} | \ \overline{\mathsf{DiffAdv}}^D] + \Pr\left[\exists\ i \notin J \text{ s.t. } \left(\mathsf{DiffAdv}^D_i\right)\right].$$

We will show that each of these terms is bounded by a negligible function.

Lemma 2.

$$\Pr[\mathsf{BadTr} | \ \overline{\mathsf{DiffAdv}}^D] \leq \mathsf{negl}(\lambda)$$

Proof. The proof of this lemma follows from Chernoff bounds. Fix any $i \in [n] \setminus J$. Let us consider the probability that the output of Trace algorithm includes i, given that $\mathsf{DiffAdv}^D_{i,j}$ does not occur for any j. Note that the tracing algorithm includes i in $\mathcal{J}$ if the estimates $\hat{p}_{i-1,j}$ and $\hat{p}_{i,j}$ differ by at least $\epsilon/4n$ for some $j \in [n]$.

Let $X_{a,j,\ell}$ be the random variable that is 1 if $D(\mathsf{ct}_{a,j,\ell}) = b_{a,j,\ell}$ for $a \in \{i-1, i\}$, $j \in [n]$, $\ell \in \{1, \ldots, W\}$, and let $Z_{i,j,\ell} = X_{i-1,j,\ell} - X_{i,j,\ell}$. Then, $\hat{p}_{a,j} = (\Sigma_{\ell \in [W]} X_{a,j,\ell})/W$ and $\nu_{i,j} = \mathbb{E}[Z_{i,j,\ell}] = p^D_{i-1,j} - p^D_{i,j}$.

Since the $Z_{i,j,\ell}$s are independent samples, using Chernoff bounds, we get that $\Pr[\Sigma_\ell Z_{i,j,\ell}/W > 2\nu_{i,j}] \leq 2^{-O(\lambda)}$. Using this, we get that, for every $i \in [n] \setminus J$, $\Pr[\mathsf{BadTr} \wedge i \in \mathcal{J} | \ \overline{\mathsf{DiffAdv}}^D] \leq n \cdot 2^{-O(\lambda)}$, by a union bound over all $j \in [n]$. By another union bound, we get that

$$\Pr[\mathsf{BadTr} | \ \overline{\mathsf{DiffAdv}}^D] \leq n^2 \cdot 2^{-O(\lambda)}.$$

□

Lemma 3. *If* PSBEG *is a secure PSBE scheme satisfying adaptive q-bounded decoder-based set hiding for* $q = W(n^2 - 1)$*, then, for all* $\lambda \in \mathbb{N}$*, for every* PPT *adversary* $\mathcal{A}$*, every non-negligible function* $\epsilon = \epsilon(\lambda)$*,*

$$\Pr\left[\exists\ i \notin J \text{ s.t. } \left(\mathsf{DiffAdv}^D_i\right)\right] \leq \mathsf{negl}(\lambda)$$

Proof. Say for contradiction, the above event $\exists\ i \notin J$ s.t. $\left(\mathsf{DiffAdv}_i^D\right)$ occurs with probability $\delta(\lambda)$. We will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the decoder-based set-hiding property of PSBEG.

$\mathcal{B}$ first receives C (or n, t if threshold access structure) from $\mathcal{A}$, and forwards this to its PSBEG challenger. It then receives the PSBEG public key pk, which it forwards to $\mathcal{A}$. Next, $\mathcal{B}$ chooses an index $i \leftarrow\$ \{1, \ldots, n\}$ (essentially $\mathcal{B}$ is trying to guess the honest party that $\mathcal{A}$ will try to falsely blame). Then, for every secret key query that $\mathcal{A}$ makes, $\mathcal{B}$ simply forwards it to its challenger, and relays the response. Eventually, $\mathcal{A}$ responds with a decoder box D, and messages μ_0, μ_1.

$\mathcal{B}$ additionally guesses $j \leftarrow\$ [n]$, as the iteration for which $\mathsf{DiffAdv}_{i,j}^D$ occurs. It then runs $j-1$ iterations of the Trace algorithm – for each step where it needs a secret-key PSBEG ciphertext, it queries its EncSK oracle. It computes the sets $\mathcal{J}_1, \ldots, \mathcal{J}_j$ exactly as in the Trace algorithm. It then samples $b' \leftarrow\$ \{0,1\}$ and returns $D, \mu_{b'}$ along with $(i-1, \mathcal{J}_j), (i, \mathcal{J}_j)$ to its challenger if $i > 1$, otherwise it returns $D, \mu_{b'}, (0, \emptyset), (1, \mathcal{J}_j)$.

We argue that if the event $\mathsf{DiffAdv}_i$ occurs for any $i \in [n] \setminus J$, and if $\mathcal{B}$ guesses i, j, b' correctly (this happens with probability at least $1/2n^2$), then $\mathcal{B}$ will win its game. To see why, observe that if $\mathcal{B}$ guesses correctly, $\mathcal{A}$ will never query the key for party i (all other key queries are allowed since for any $i' \neq i$, $i' \notin \mathcal{S}_0 \Delta \mathcal{S}_1$ as defined in the set-hiding definition). Additionally, if $\mathcal{B}$ guesses j correctly, we show below that the box D must be able to distinguish between encryptions of $\mu_{b'}$ to $(i-1, \mathcal{J}_j)$ and $(i, \mathcal{J}_j)$ (and between $(0, \emptyset)$ and $(1, \mathcal{J}_j)$ for the case of $i = 1$):

Let $p_{b,i'}^D = \Pr[D(\mathsf{ct}) = b : \mathsf{ct} \leftarrow\$ \mathsf{EncSK}(\mathsf{msk}, \mu_b, i', \mathcal{J}_j)]$, where the probability is taken over randomness of D and random coins of the encryption algorithm (for $i' = 0$, we replace $\mathcal{J}_j$ by $\emptyset$ but the following argument applies to this case as well). Recall that if $\mathcal{B}$ guessed correctly, we have that i is honest and $\mathsf{DiffAdv}_{i,j}$ occurs. Therefore, we get that for i, j guessed by $\mathcal{B}$,

$$\Pr[i \notin J \wedge (p_{0,i-1}^D + p_{1,i-1}^D)/2 - (p_{0,i}^D + p_{1,i}^D)/2 > \epsilon/8n] \geq \delta(\lambda)/n^2.$$

This also means that there exists a bit b such that

$$\Pr[i \notin J \wedge (p_{b,i-1}^D - p_{b,i}^D \geq \epsilon/8n)] \geq \delta(\lambda)/n^2.$$

Now since $\mathcal{B}$ guesses b' uniformly at random, we have that $\mathcal{B}$ wins its game with probability $\delta/2n^2$. The lemma now follows from adaptive q-bounded decoder-based set-hiding of PSBEG. □

Correct Trace Probability. Now we show, whenever the adversary outputs a good decoder box, then with all but negligible probability, we will find at least one traitor in each iteration of the tracing algorithm. Moreover, we never find the same traitor in two different iterations. Combining the two, we get that the tracing algorithm will stop in upto n iterations, once it finds an authorized set of traitors.

Let us start with some notation. Let E_j be the event that no party was found in the jth iteration of the tracing algorithm i.e. $\mathcal{J}_j' = \emptyset$. Additionally, for $j < j'$, let $\mathsf{E}_{j,j'}$ be the event that the same party was found in iterations $j \neq j'$, i.e. $\mathcal{J}_{j'}' \cap \mathcal{J}_j' \neq \emptyset$. Let $\mathsf{E}_{overlap}$ denote the event that $\mathsf{E}_{j,j'}$ occurs for some $j < j'$, i.e. $\mathsf{E}_{overlap} := \bigvee_{j<j'} \mathsf{E}_{j,j'}$. We will now bound the probability of all these events:

Theorem 14. *Assuming the* PSBEG *scheme achieves indistinguishability and adaptive q-query decoder-based message-hiding for $q = W(n^2 - 1)$, For every* PPT *adversary $\mathcal{A}$, polynomial $p(\cdot)$, and every non-negligible function $\epsilon(\cdot)$ such that $\epsilon(\lambda) > 1/p(\lambda)$,*

$$\Pr[\overline{\mathsf{E}}_j \text{ for all } j] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda),$$

Here j is upper bounded by the number of iterations run in the Trace *algorithm, and* GoodDec *is the event that the box D output by $\mathcal{A}$ is good with respect to $\epsilon(\lambda)$.*

Proof. First, the event GoodDec implies that $p^D_{nrml} \geq 1/2 + \epsilon$. Then, by indistinguishability of PSBEG, we have that $p^D_{nrml} - p^D_{0,j} \leq \mathsf{negl}(\lambda)$ for all $j \in [n]$, since for $i = 0$, we always encrypt to $(0, \emptyset)$.

Additionally, we argue that by adaptive q-query decoder-based message-hiding of PSBEG, $p^D_{n,j}$ must be negligible for all iterations j run by the tracing algorithm. We formally prove this below. Let $\mathsf{E}^D_{msg,j}$ denote the event that $p^D_{n,j} > 1/2 + \delta(\lambda)$ where δ is a non-negligible function. Let $\mathsf{E}^D_{msg} = \bigvee_{j\in[n]} \mathsf{E}^D_{msg,j}$.

Lemma 4. *Assuming* PSBEG *satisfies adaptive q-query decoder-based message-hiding, for every* PPT *adversary $\mathcal{A}$,*

$$\Pr[\mathsf{E}^D_{msg}] = \mathsf{negl}(\lambda).$$

Proof. To prove this, we will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks adaptive q-query decoder-based message-hiding of PSBEG. The game starts with $\mathcal{B}$ receiving C from $\mathcal{A}$, which it forwards to its challenger. $\mathcal{B}$ gets back $\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n$ from its challenger. $\mathcal{B}$ forwards pk to $\mathcal{A}$, and uses $\{\mathsf{sk}_i\}$ to respond to secret key queries made by $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs a box D along with messages μ_0, μ_1.

$\mathcal{B}$ then guesses the index j^* for which D can break message-hiding (i.e. the event E^D_{msg,j^*} occurs). To compute $\mathcal{J}_{j^*}$, $\mathcal{B}$ runs the first $j^* - 1$ iterations of the Trace algorithm. In particular, for each $j \in [j^* - 1]$, for $i \in \{0, \ldots, n\}$, and for $\ell \in [W]$, $\mathcal{B}$ samples a bit $b_{i,j,\ell}$, and computes a ciphertext $\mathsf{ct}_{i,j,\ell}$ by making an EncSK query to its challenger. It then computes $\hat{p}_{i,j}$, $\mathcal{J}'_j$, and updates the set $\mathcal{J}$ as in the Trace algorithm. Then, $\mathcal{B}$ outputs $D, \mathcal{J}_{j^*}$ (where $\mathcal{J}_j$ is the value of $\mathcal{J}$ at the beginning of the jth iteration) along with μ_0, μ_1.

We argue that $\mathcal{B}$ wins its game with probability $\frac{1}{n} \cdot \Pr[\mathsf{E}^D_{msg}]$. To see why, observe that if the event E^D_{msg} occurs, then, there is at least one value j for which the event $\mathsf{E}^D_{msg,j}$ occurs. $\mathcal{B}$ guesses this j^* correctly with probability at least $1/n$. Moreover, by construction of the tracing algorithm, $\mathcal{J}_{j^*}$ must be unauthorized, since otherwise the algorithm would have terminated. Hence in the event E^D_{msg} and if $\mathcal{B}$ guesses correctly, it does indeed win the message-hiding game. Then, adaptive q-query decoder-based message-hiding implies that $\Pr[\mathsf{E}^D_{msg}]$ must be negligible. □

So we have, for all iterations $j \leq n$ run by the tracing algorithm, $p^D_{nrml} - p^D_{0,j} \leq \mathsf{negl}(\lambda)$ i.e. $p^D_{0,j} \geq 1/2 + \epsilon(\lambda) - \mathsf{negl}(\lambda)$, and with overwhelming probability, $p^D_{n,j} \leq 1/2 + \mathsf{negl}(\lambda)$. This implies $p^D_{0,j} - p^D_{n,j} \geq \epsilon - \mathsf{negl}(\lambda) \geq \epsilon/2$. Hence, for every $j \leq n$, there must be at least one index $i^*_j \in [n]$ such that $p^D_{i^*_j - 1,j} - p^D_{i^*_j,j} > \epsilon/2n$. Then, using Chernoff bounds, we get that for all $j \in [n]$,

$$\Pr[\hat{p}^D_{i^*_j-1,j} - \hat{p}^D_{i^*_j,j} < \epsilon/4n] \leq 2^{-O(\lambda)} = \mathsf{negl}(\lambda).$$

This implies that,

$$\Pr[\overline{\mathsf{E}}_j \text{ for all } j \in [n]] \geq (1 - \mathsf{negl}(\lambda)) \cdot \Pr[\mathsf{GoodDec}] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda).$$

Hence, if D is a good box, we will find at least one party in the jth iteration, i.e. $\mathcal{J}'_j \neq \emptyset$ for all iterations executed by the tracer. This proves the theorem. □

Lastly, we prove that we never find the same traitor in two different iterations of the tracing algorithm. Let $\mathsf{E}^D_{j,j'}$ denote the event that for some $i \in \mathcal{J}'_{j'}$, $p^D_{i-1,j} - p^D_{i,j} > \epsilon/8n$. Let $\mathsf{E}^D_{overlap}$ denote the event that for some $j < j'$, event $\mathsf{E}^D_{j,j'}$ occurs.

Theorem 15. *Assuming* PSBEG *achieves adaptive q-query decoder-based set-hiding for $q = W(n^2-1)$, then, for every* PPT *adversary $\mathcal{A}$,*

$$\Pr[\mathsf{E}^D_{overlap}] \leq \mathsf{negl}(\lambda).$$

Proof. We will construct an adversary $\mathcal{B}$ that breaks set-hiding using $\mathcal{A}$ in the event $\mathsf{E}^D_{overlap}$. The game starts with $\mathcal{A}$ sending C to $\mathcal{B}$ which it forwards to its challenger. $\mathcal{B}$ gets pk from its challenger, which it sends to $\mathcal{A}$. $\mathcal{B}$ guesses iterations $j' < j \in [n]$ for which the event $\mathsf{E}^D_{j,j'}$ occurs, and guesses a party $i^* \in [n]$ such that $i^* \in \mathcal{J}'_{j'}$ and $p^D_{i^*-1,j} - p^D_{i^*,j} > \epsilon/8n$.

Then, for every $\mathsf{skO}(i)$ query made by $\mathcal{A}$, $\mathcal{B}$ simply forwards the query to its challenger, and relays the response back to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs a decoder D along with μ_0, μ_1. $\mathcal{B}$ then runs $j-1$ iterations of the tracing algorithm: herein, it queries its challenger for ciphertexts encrypted via EncSK. It then computes $\mathcal{J}_1, \ldots, \mathcal{J}_j$ as in the algorithm. $\mathcal{B}$ samples $b' \leftarrow\$ \{0,1\}$ and returns $D, \mu_{b'}$ along with $(\mathcal{J}_j, i^*-1), (\mathcal{J}_j, i^*)$ to its challenger (note if $i^* = 1$, $\mathcal{B}$ will return $(\emptyset, 0)$ and $(\mathcal{J}_j, 1)$ and the remaining analysis works directly for this case as well).

We argue that if the event $\mathsf{E}^D_{overlap}$ occurs and if $\mathcal{B}$ guesses j, j', i^*, b correctly, then $\mathcal{B}$ wins its set-hiding game. To see why, first observe that if $\mathcal{B}$ guesses correctly (i.e. event $\mathsf{E}^D_{j',j}$ occurs, and $i^* \in \mathcal{J}'_{j'}$ and $p^D_{i^*-1,j} - p^D_{i^*,j} > \epsilon/8n$), then $i^* \in \mathcal{J}_j$ by construction, since $\mathcal{J}_j$ is the superset of traitors $\mathcal{J}'_k$ caught in iteration k for all $k < j$, in particular it is the superset of $\mathcal{J}'_{j'}$. Additionally, $\mathcal{B}$ does not make any invalid key queries, since for all $i \in [n]$, $i \in \{i^*, \ldots, n\} \cup \mathcal{J}_j$ if and only if $i \in \{i^*+1, \ldots, n\} \cup \mathcal{J}_j$ since $i^* \in \mathcal{J}_j$. Also to run the Trace algorithm, $\mathcal{B}$ makes a maximum of $W \cdot (n+1) \cdot (n-1)$ EncSK queries, which it is allowed to make. Lastly, given that $p^D_{i^*-1,j} - p^D_{i^*,j} > \epsilon/8n$ (if $\mathcal{B}$ guessed correctly), we can use analysis similar to proof of Lemma 3, to argue that there exists a bit b such that the behavior of D differs between encryption of μ_b to $(i^*-1, \mathcal{J}_j)$ vs $(i^*, \mathcal{J}_j)$.

Since in the event $\mathsf{E}^D_{overlap}$, there is at least one pair j, j' such that for some $i \in \mathcal{J}'_{j'}$, $p^D_{i-1,j} - p^D_{i,j} > \epsilon/8n$, $\mathcal{B}$ will guess j, j' correctly with probability $\geq 1/n^2$, and guess i^* correctly with probability $1/n$. Lastly, it guesses b correctly with probability $1/2$. Hence, we get that the advantage of $\mathcal{B}$ is at least $\frac{1}{2n^3} \cdot \Pr[\mathsf{E}^D_{overlap}]$. Adaptive q-query decoder-based set-hiding for $q = W(n^2-1)$ implies that the event $\mathsf{E}^D_{overlap}$ can occur only with negligible probability. □

Lemma 5.

$$\Pr[\mathsf{E}_{overlap} | \overline{\mathsf{E}^D_{overlap}}] \leq \mathsf{negl}(\lambda)$$

Proof. The proof directly follows the proof of Lemma 2. Essentially, since $\overline{\mathsf{E}^D_{overlap}}$ implies that for all $j < j'$, for all $i \in \mathcal{J}'_{j'}$, $p^D_{i-1,j} - p^D_{i,j} \leq \epsilon/8n$. Then, by Chernoff bounds, we get that the estimated probabilities $\hat{p}^D_{i-1,j}$ and $\hat{p}^D_{i,j}$ cannot be too far apart, meaning i will not be included in $\mathcal{J}'_j$ except with negligible probability. □

Combining Theorem 15 and Lemma 5, we get that $\Pr[\mathsf{E}_{overlap}] \leq \mathsf{negl}(\lambda)$, meaning we never blame the same party in two different iterations of the tracing algorithm.

Combining all the above, we get that if D is indeed a good box, we will trace one unique traitor in each iteration with overwhelming probability. More formally, we get,

$$\Pr[\mathsf{GoodTr}] \geq 1 - \Pr[\mathsf{E}_j \text{ for some } j] - \Pr[\mathsf{E}_{overlap}] \geq \Pr[\mathsf{GoodDec}] - \mathsf{negl}(\lambda).$$

And by the false trace analysis, none of the traced parties is corrupt. More formally, $\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda)$.

Hence, we will find an authorized subset of parties by the end of the tracing algorithm (note since the set of all n parties must be authorized for any monotone access structure, we will indeed find an authorized subset in upto n iterations). □

On Traitor Tracing from PSBEG with Selective Security We now show that we can use the same compiler as above to construct TTG from a PSBEG scheme with 0-query selective security by using complexity leveraging. Correctness and semantic security remain the same as before, so we show traceability in Theorem 16.

Theorem 16. *For every* PPT *adversary* $\mathcal{A}$ *for traceability of* TTG*, there exist* PPT *adversaries* $\hat{\mathcal{B}}, \mathcal{B}, \mathcal{B}_1, \mathcal{B}_2$ *such that,*

$$\mathsf{Adv}^{\mathrm{trac}}_{\mathcal{A},\mathsf{TTG}}(\lambda) \leq \begin{pmatrix} \mathsf{negl}(\lambda) + 2n \cdot \binom{n}{t-1} \cdot \mathsf{Adv}^{\mathrm{sel,0,dec,set-hiding}}_{\hat{\mathcal{B}},\mathsf{PSBEG}}(\lambda) + \\ \mathsf{Adv}^{\mathrm{indist}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda) + \\ \binom{n}{t-1} \cdot \mathsf{Adv}^{\mathrm{sel,0,dec,msg-hiding}}_{\mathcal{B}_1,\mathsf{PSBEG}}(\lambda) + \\ 2n\binom{n}{t-2} \cdot \mathsf{Adv}^{\mathrm{sel,0,dec,set-hiding}}_{\mathcal{B}_2,\mathsf{PSBEG}}(\lambda) \end{pmatrix},$$

where $t-1$ *denotes the size of the largest unauthorized set in the access structure for* TTG*.*

Proof (of Theorem 16). We use the same notation as the proof of Theorem 12.

False Trace Probability.

Theorem 17. *For every* PPT *adversary* $\mathcal{A}$*, there exists another* PPT *adversary* $\mathcal{B}$ *such that,*

$$\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda) + 2n \cdot \binom{n}{t-1} \cdot \mathsf{Adv}^{\mathrm{sel,0,dec,set-hiding}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda),$$

wherein $\mathsf{Adv}^{\mathrm{sel,0,dec,set-hiding}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda)$ *is the advantage of* $\mathcal{B}$ *in the selective,* 0*-query decoder-based set-hiding game.*

Proof. The proof follows the proof of Theorem 13: we write the probability BadTr as a sum over conditionals, and use Chernoff bounds to show that the probability conditioned on $\overline{\mathsf{DiffAdv}}^D$ is negligible. The following lemma now completes the proof:

Lemma 6. *For every* PPT *adversary* $\mathcal{A}$*, there exists another* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\mathrm{sel,0,dec,set-hiding}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda) \geq \frac{1}{(2n \cdot \binom{n}{t-1})} \Pr\left[\exists\ i \notin J\ s.t.\ \left(\mathsf{DiffAdv}^D_i\right)\right]$$

Proof. Say for contradiction, the above event $\exists\ i \notin J$ s.t. $\left(\mathsf{DiffAdv}^D_i\right)$ occurs with probability $\delta(\lambda)$. We will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the decoder-based set-hiding property of PSBEG.

$\mathcal{B}$ first receives C (or n, t if threshold access structure) from $\mathcal{A}$, and forwards this to its PSBEG challenger. It then receives the PSBEG public key pk, which it forwards to $\mathcal{A}$. Next, $\mathcal{B}$ chooses an index $i \leftarrow\$ \{1, \ldots, n\}$ (essentially $\mathcal{B}$ is trying to guess the honest party that $\mathcal{A}$ will try to falsely

blame). Then, for every secret key query that $\mathcal{A}$ makes, $\mathcal{B}$ simply forwards it to its challenger, and relays the response. Eventually, $\mathcal{A}$ responds with a decoder box D, and messages μ_0, μ_1.

Now, instead of guessing the iteration $j \in [n]$ and running the Trace algorithm to get $\mathcal{J}_j$, $\mathcal{B}$ will directly guess the set $\mathcal{J}_j$, as a random subset of $[n]$ of size $t-1$ (recall $\mathcal{J}_j$ must be unauthorized). This is $\mathcal{B}$'s guess for $\mathcal{J}_j$ where $\mathsf{DiffAdv}^D_{i,j}$ occurs for iteration j.

The rest of the proof follows the proof of Lemma 3: $\mathcal{B}$ samples $b' \leftarrow\!\!\$\ \{0,1\}$ and returns $D, \mu_{b'}$ along with $(i-1, \mathcal{J}_j), (i, \mathcal{J}_j)$ to its challenger if $i > 1$, otherwise it returns $D, \mu_{b'}, (0, \emptyset), (1, \mathcal{J}_j)$.

We argue that if the event $\mathsf{DiffAdv}_i$ occurs for any $i \in [n] \setminus J$, and if $\mathcal{B}$ guesses $i, \mathcal{J}_j, b'$ correctly (this happens with probability at least $1/(2n \cdot \binom{n}{t-1})$), then $\mathcal{B}$ will win its game. To see why, observe that if $\mathcal{B}$ guesses correctly, $\mathcal{A}$ will never query the key for party i (all other key queries are allowed since for any $i' \neq i$, $i' \notin \mathcal{S}_0 \Delta \mathcal{S}_1$ as defined in the set-hiding definition). Additionally, if $\mathcal{B}$ guesses $\mathcal{J}_j$ correctly, we show below that the box D must be able to distinguish between encryptions of $m_{b'}$ to $(i-1, \mathcal{J}_j)$ and $(i, \mathcal{J}_j)$:

Let $p^D_{b,i'} = \Pr[D(\mathsf{ct}) = b : \mathsf{ct} \leftarrow\!\!\$\ \mathsf{EncSK}(\mathsf{msk}, \mu_b, i', \mathcal{J}_j)]$, where the probability is taken over randomness of D and random coins of the encryption algorithm (for $i' = 0$, we replace $\mathcal{J}_j$ by $\emptyset$ but the following argument applies to this case as well). Recall that if $\mathcal{B}$ guessed correctly, we have that i is honest and $\mathsf{DiffAdv}_{i,j}$ occurs for the set $\mathcal{J}_j$. Therefore, we get that for $i, \mathcal{J}_j$ guessed by $\mathcal{B}$,

$$\Pr[i \notin J \wedge (p^D_{0,i-1} + p^D_{1,i-1})/2 - (p^D_{0,i} + p^D_{1,i})/2 > \epsilon/8n] \geq \delta(\lambda)/(n \cdot \binom{n}{t-1}).$$

This also means that there exists a bit b such that

$$\Pr[i \notin J \wedge (p^D_{b,i-1} - p^D_{b,i} \geq \epsilon/8n)] \geq \delta(\lambda)/(n \cdot \binom{n}{t-1}).$$

Now since $\mathcal{B}$ guesses b' uniformly at random, we have that $\mathcal{B}$ wins its game with probability $\delta/(2n \cdot \binom{n}{t-1})$. This proves the lemma. □

Correct Trace Probability. Similar to the proof of Theorem 12, we will prove that we find at least one traitor in each iteration of the tracing algorithm, and that we never find the same traitor in two different iterations.

Theorem 18. *For every* PPT *adversary* $\mathcal{A}$*, polynomial* $p(\cdot)$*, every* $j \in [n]$*, and every non-negligible function* $\epsilon(\cdot)$ *such that* $\epsilon(\lambda) > 1/p(\lambda)$*,*

$$\Pr[\mathsf{E}_j \text{ for some } j | \mathsf{GoodDec}] \leq \mathsf{Adv}^{\mathrm{indist}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda) + \binom{n}{t-1} \cdot \mathsf{Adv}^{\mathrm{sel,0,dec,msg-hiding}}_{\mathcal{B}_1,\mathsf{PSBEG}}(\lambda) + \mathsf{negl}(\lambda),$$

where $\mathsf{GoodDec}$ *is the event that the box* D *output by* $\mathcal{A}$ *is good with respect to* $\epsilon(\lambda)$*, and* j *is upper bounded by the number of iterations in the* Trace *algorithm.*

Proof. First, the event $\mathsf{GoodDec}$ implies that $p^D_{nrml} \geq 1/2 + \epsilon$. Then, by indistinguishability of PSBEG, we have that $p^D_{nrml} - p^D_{0,j} \leq \mathsf{negl}(\lambda)$ for all $j \in [n]$, since for $i = 0$, we always encrypt to $(0, \emptyset)$. More formally, if this is not the case, then we can construct an adversary $\mathcal{B}$ to break indistinguishability of PSBEG.

Additionally, we argue that by selective decoder-based message-hiding of PSBEG, $p^D_{n,j}$ must be $1/2$ plus negligible for all j. We formally prove this below. Let $\mathsf{E}^D_{msg,j}$ denote the event that $p^D_{n,j} > 1/2 + \delta(\lambda)$ where δ is a non-negligible function. Let $\mathsf{E}^D_{msg} = \bigvee_{j \in [n]} \mathsf{E}^D_{msg,j}$.

Lemma 7. *For every* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}_1$ *such that,*

$$\mathsf{Adv}^{\mathrm{sel},0,\mathrm{dec},\mathrm{msg-hiding}}_{\mathcal{B}_1,\mathsf{PSBEG}}(\lambda) \geq \frac{1}{\binom{n}{t-1}} \cdot \Pr[\mathsf{E}^D_{msg}],$$

where $\mathsf{Adv}^{\mathrm{sel},0,\mathrm{dec},\mathrm{msg-hiding}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda)$ *is the advantage of* $\mathcal{B}$ *in the selective,* 0*-query decoder-based message hiding game.*

Proof. We follow the proof of Lemma 4: We use $\mathcal{A}$ to construct $\mathcal{B}_1$ that breaks selective decoder-based message-hiding of PSBEG. The game starts with $\mathcal{B}_1$ receiving C from $\mathcal{A}$, which it forwards to its challenger. $\mathcal{B}_1$ gets back $\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n$ from its challenger. $\mathcal{B}_1$ forwards pk to $\mathcal{A}$, and uses $\{\mathsf{sk}_i\}$ to respond to secret key queries made by $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs a box D along with messages μ_0, μ_1.

Then, instead of guessing iteration j^* and running the Trace algorithm, $\mathcal{B}_1$ simply guesses a set $\mathcal{J}$ randomly from all sets of size $t-1$, as its guess for $\mathcal{J}_{j^*}$. Then, $\mathcal{B}_1$ outputs $D, \mathcal{J}$ along with μ_0, μ_1.

We argue that $\mathcal{B}_1$ wins its game with probability $\frac{1}{\binom{n}{t-1}} \cdot \Pr[\mathsf{E}^D_{msg}]$. To see why, observe that if the event E^D_{msg} occurs, then, there is at least one value j for which the event $\mathsf{E}^D_{msg,j}$ occurs. $\mathcal{B}$ guesses the set $\mathcal{J}_{j^*}$ correctly with probability at least $1/\binom{n}{t-1}$, since, by construction of the tracing algorithm, $\mathcal{J}_{j^*}$ must be unauthorized. Hence in the event E^D_{msg} and if $\mathcal{B}_1$ guesses correctly, it does indeed win the message-hiding game. □

So we have, for all iterations $j \in [n]$, $p^D_{nrml} - p^D_{0,j} \leq \mathsf{negl}(\lambda)$ i.e. $p^D_{0,j} \geq 1/2 + \epsilon(\lambda) - \mathsf{negl}(\lambda)$, and $p^D_{n,j} \leq 1/2 + \mathsf{negl}(\lambda)$ by the reduction to message-hiding of PSBEG. This implies $p^D_{0,j} - p^D_{n,j} \geq \epsilon - \mathsf{negl}(\lambda) \geq \epsilon/2$. Hence, there must be at least one index i^*_j such that $p^D_{i^*_j-1,j} - p^D_{i^*_j,j} > \epsilon/2n$. Then, using Chernoff bounds, we get that for all $j \in [n]$,

$$\Pr[\hat{p}^D_{i^*_j-1,j} - \hat{p}^D_{i^*_j,j} < \epsilon/4n] \leq 2^{-O(\lambda)} = \mathsf{negl}(\lambda).$$

Combining it all, we get the theorem. Informally, if D is a good box, we will find at least one party in the jth iteration, i.e. $\mathcal{J}'_j \neq \emptyset$ for all $j \in [n]$.

Next we prove that we never find the same traitor in two different iterations of the tracing algorithm. Let $\mathsf{E}^D_{j,j'}$ denote the event that for some $i \in \mathcal{J}'_{j'}$, $p^D_{i-1,j} - p^D_{i,j} > \epsilon/8n$. Let $\mathsf{E}^D_{overlap}$ denote the event that for some $j < j'$, event $\mathsf{E}^D_{j,j'}$ occurs.

Theorem 19. *For every* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}_2$ *such that*

$$\mathsf{Adv}^{\mathrm{sel},0,\mathrm{dec},\mathrm{set-hiding}}_{\mathcal{B}_2,\mathsf{PSBEG}}(\lambda) \geq \frac{1}{2n\binom{n}{t-2}} \Pr[\mathsf{E}^D_{overlap}].$$

Proof. We follow the proof of Theorem 15. We will construct $\mathcal{B}_2$ using $\mathcal{A}$. The game starts with $\mathcal{A}$ sending C to $\mathcal{B}_2$ which it forwards to its challenger. $\mathcal{B}_2$ gets pk from its challenger, which it sends to $\mathcal{A}$. Then, instead of guessing j, j', i^*, $\mathcal{B}_2$ directly guesses $i^* \in [n]$ and a set $\mathcal{J} \subseteq [n]$, wherein $i^* \in \mathcal{J}$, and $\mathcal{J}$ is its guess for $\mathcal{J}_j$.

Then, for every $\mathsf{skO}(i)$ query made by $\mathcal{A}$, $\mathcal{B}_2$ simply forwards the query to its challenger, and relays the response back to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs a decoder D along with μ_0, μ_1. Note $\mathcal{B}_2$ does not need to run the tracing algorithm since it just guessed the set $\mathcal{J}$. $\mathcal{B}_2$ samples $b' \leftarrow\!\$\ \{0,1\}$ and

returns $D, \mu_{b'}$ along with $(\mathcal{J}, i^* - 1), (\mathcal{J}, i^*)$ to its challenger (note if $i^* = 1$, $\mathcal{B}_2$ will return $(\emptyset, 0)$ and $(\mathcal{J}, 1)$ and the remaining analysis works directly for this case as well).

We argue that if the event $\mathsf{E}^D_{overlap}$ occurs and if $\mathcal{B}_2$ guesses $\mathcal{J}, i^*, b$ correctly, then $\mathcal{B}_2$ wins its set-hiding game. To see why, first observe that if $\mathcal{B}_2$ guesses correctly (i.e. event $\mathsf{E}^D_{j',j}$ occurs, and $i^* \in \mathcal{J}'_{j'}$, $\mathcal{J}_j = \mathcal{J}$ and $p^D_{i^*-1,j} - p^D_{i^*,j} > \epsilon/8n$), then $i^* \in \mathcal{J}_j$ by construction. Additionally, $\mathcal{B}_2$ does not make any invalid key queries, since for all $i \in [n]$, $i \in \{i^*, \ldots, n\} \cup \mathcal{J}_j$ if and only if $i \in \{i^* + 1, \ldots, n\} \cup \mathcal{J}_j$ since $i^* \in \mathcal{J}_j$. Also $\mathcal{B}_2$ makes 0 EncSK queries. Lastly, given that $p^D_{i^*-1,j} - p^D_{i^*,j} > \epsilon/8n$, we can use analysis similar to proof of Lemma 3, to argue that there exists a bit b such that the behavior of D differs between encryption of μ_b to $(i^* - 1, \mathcal{J}_j)$ vs $(i^*, \mathcal{J}_j)$.

Since in the event $\mathsf{E}^D_{overlap}$, there is at least one pair j, j' such that for some $i \in \mathcal{J}'_{j'}$, $p^D_{i-1,j} - p^D_{i,j} > \epsilon/8n$, $\mathcal{B}_2$ will guess $\mathcal{J}$ and i correctly with probability $\geq 1/(n \cdot \binom{n}{t-2})$ (note $\mathcal{J}$ has i, so $\mathcal{B}$ is only guessing the remaining $t-2$ elements in $\mathcal{J}$). Lastly, it guesses b correctly with probability $1/2$. Hence, we get that the advantage of $\mathcal{B}_2$ is at least $\frac{1}{2n\binom{n}{t-2}} \cdot \Pr[\mathsf{E}^D_{overlap}]$. □

Lemma 8.

$$\Pr[\mathsf{E}_{overlap} | \overline{\mathsf{E}^D_{overlap}}] \leq \mathsf{negl}(\lambda)$$

Proof. The proof directly follows the proof of Lemma 2. □

Combining Theorem 15 and Lemma 8, we get that $\Pr[\mathsf{E}_{overlap}] \leq \mathsf{negl}(\lambda)$, meaning we never blame the same party in two different iterations of the tracing algorithm.

Combining all the above, we get that if D is indeed a good box, we will trace one unique party in each iteration with overwhelming probability, and by the false trace analysis, none of the traced parties is corrupt. Hence, we will find an authorized subset of parties by the end of the tracing algorithm (note since the set of all n parties must be authorized for any monotone access structure, we will indeed find an authorized subset in upto n iterations). More formally:

$$\begin{aligned}
\Pr[\overline{\mathsf{GoodTr}}] &\leq \Pr[\overline{\mathsf{GoodDec}}] + \Pr[\mathsf{GoodDec}] \cdot (\Pr[\mathsf{E}_j \text{ for some } j | \mathsf{GoodDec}] + \Pr[\mathsf{E}_{overlap} | \mathsf{GoodDec}]) \\
&\leq \Pr[\overline{\mathsf{GoodDec}}] + \Pr[\mathsf{GoodDec}] \cdot \begin{pmatrix} \mathsf{Adv}^{\mathsf{indist}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda) + \\ \binom{n}{t-1} \cdot \mathsf{Adv}^{\mathsf{sel},0,\mathsf{dec},\mathsf{msg}-\mathsf{hiding}}_{\mathcal{B}_1,\mathsf{PSBEG}}(\lambda) + \\ 2n\binom{n}{t-2} \cdot \mathsf{Adv}^{\mathsf{sel},0,\mathsf{dec},\mathsf{set}-\mathsf{hiding}}_{\mathcal{B}_2,\mathsf{PSBEG}}(\lambda) + \\ \mathsf{negl}(\lambda) \end{pmatrix}
\end{aligned}$$

Simplifying the above, we get,

$$\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \begin{pmatrix} \mathsf{Adv}^{\mathsf{indist}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda) + \\ \binom{n}{t-1} \cdot \mathsf{Adv}^{\mathsf{sel},0,\mathsf{dec},\mathsf{msg}-\mathsf{hiding}}_{\mathcal{B}_1,\mathsf{PSBEG}}(\lambda) + \\ 2n\binom{n}{t-2} \cdot \mathsf{Adv}^{\mathsf{sel},0,\mathsf{dec},\mathsf{set}-\mathsf{hiding}}_{\mathcal{B}_2,\mathsf{PSBEG}}(\lambda) + \\ \mathsf{negl}(\lambda) \end{pmatrix}.$$

Combining the above with Theorem 17, we get the theorem. □

6.4 Limited Traitor Tracing for General Access Structures from Adaptive 0-query Decoder-based PSBEG

In this section, we present a different generic compiler to build traitor tracing from PSBEG that only achieves adaptive 0-query decoder-based security. However, we can only achieve limited tracing with

$\underline{\mathsf{KeyGen}(1^\lambda, C, 1^{1/\epsilon}) \to (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) :}$

1. Let $\delta \leftarrow (1/2 - \epsilon)/(1/2 - 2/\sqrt{\lambda})$ and $Y \leftarrow t/(1-\delta)$. // $t = 1 + t'$, where $t' \le n$ is the size of the largest unauthorized subset w.r.t. C
2. For $j \in \{1, \ldots, Y\}$: Let $(\mathsf{pk}_j, \mathsf{msk}_j, \{\mathsf{sk}_{j,i}\}_{i\in[n]}) \leftarrow\!\$\ \mathsf{PSBEG.KeyGen}(1^\lambda, C)$.
3. Output $(\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_Y), \mathsf{tk} := (\mathsf{msk}_1, \ldots, \mathsf{msk}_Y), \{\mathsf{sk}_i = (\mathsf{sk}_{1,i}, \ldots, \mathsf{sk}_{Y,i})\}_{i\in[n]})$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu) \to \mathsf{ct} :}$

1. Sample $j \leftarrow\!\$\ [Y]$. Output $(j, \mathsf{PSBEG.EncPK}(\mathsf{pk}_j, \mu))$.

$\underline{\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}) \to d_i :}$

1. Parse ct as (j, ct').
2. Output $\mathsf{PSBEG.Dec}(\mathsf{sk}_{j,i}, \mathsf{ct}')$.

$\underline{\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_i\}_{i\in\mathcal{J}}) \to \mu :}$

1. Parse ct as (j, ct') and output $\mathsf{PSBEG.Combine}(\mathsf{pk}_j, \mathsf{ct}', \{d_i\}_{i\in\mathcal{J}})$.

$\underline{\mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^y, \mu_0, \mu_1) :}$

1. Let $y := 1/\epsilon$ and $W := \lambda \cdot (n \cdot y)^2$.
2. Let $\mathcal{J} := \emptyset$ and $j = 1$. While $C(x_{\mathcal{J}}) = 0$ and $j < Y$:
   (a) For $i = 0 \ldots n$
      i. Let $\mathsf{count}_{i,j} := 0$.
      ii. For $\ell = 1$ to W:
         A. Sample $b_{i,j,\ell} \leftarrow \{0, 1\}$.
         B. If $i = 0$, let $\mathsf{ct}_{i,j,\ell} \leftarrow\!\$\ \mathsf{PSBEG.EncSK}(\mathsf{msk}_j, \mu_{b_{i,j,\ell}}, i, \emptyset)$.
         C. Else if $i > 0$, then define $\mathsf{ct}_{i,j,\ell} \leftarrow\!\$\ \mathsf{PSBEG.EncSK}(\mathsf{msk}_j, \mu_{b_{i,j,\ell}}, i, \mathcal{J})$.
         D. If $D((j, \mathsf{ct}_{i,j,\ell})) = b_{i,j,\ell}$, update $\mathsf{count}_{i,j} := \mathsf{count}_{i,j} + 1$.
      iii. Let $\hat{p}_{i,j} := \mathsf{count}_{i,j}/W$.
   (b) Let $\mathcal{J}'_j := \{i \mid \hat{p}_{i-1,j} - \hat{p}_{i,j} \ge \epsilon/4n\}_{i\in[n]}$. Set $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}'_j$ and $j \leftarrow j + 1$.
3. **output** $\mathcal{J}$.

Fig. 10. Our compiler to construct a limited traitor tracing scheme $\mathsf{TTG}_{\mathsf{ltd}}$ for general access structures from PSBEG with adaptive 0-query decoder based security for the same access structure. This construction can only trace decoders with $\epsilon \ge 2/\sqrt{\lambda}$.

this compiler, meaning we can only trace boxes that work with probability $\epsilon \ge 2/\sqrt{\lambda}$. Figure 10 presents the compiler.

Correctness. Correctness follows trivially from correctness of PSBEG.

Semantic Security. Theorem 20 below proves semantic security of the resulting scheme by relying on semantic security of PSBEG.

Theorem 20. *For every* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda) = \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A},\mathsf{TTG}_{\mathsf{ltd}}}(\lambda)$$

Proof (of Theorem 20). We will use $\mathcal{A}$ to construct $\mathcal{B}$:

- $\mathcal{B}$ forwards 1^λ to $\mathcal{A}$ and receives (C, state). $\mathcal{B}$ then forwards this to its challenger.
- $\mathcal{B}$ receives $\mathsf{pk}, \mathsf{msk}$ from its challenger. $\mathcal{B}$ samples $j^* \leftarrow\!\$\ [Y]$, where Y is as defined in the scheme, and sets $\mathsf{pk}_{j^*} \leftarrow \mathsf{pk}$ and $\mathsf{msk}_{j^*} \leftarrow \mathsf{msk}$.

– $\mathcal{B}$ samples all the other instances of PSBEG on its own. In particular, for all $j \in [Y] \setminus \{j^*\}$, it samples $(\mathsf{pk}_j, \mathsf{msk}_j, \mathsf{sk}_{j,1}, \ldots, \mathsf{sk}_{j,n}) \leftarrow\$ \mathsf{PSBEG.KeyGen}(1^\lambda, C)$.
– $\mathcal{B}$ then sends $\mathsf{pk} = \{\mathsf{pk}_j\}_{j\in[Y]}, \mathsf{msk} = \{\mathsf{msk}_j\}_{j\in[Y]}$ to $\mathcal{A}$.
– For any secret key query i by $\mathcal{A}$, $\mathcal{B}$ simply queries its challenger for the secret key of i, let the response be $\mathsf{sk}_{j^*,i}$, it then outputs the full secret key to $\mathcal{A}$.
– For any partial decryption query: if it is for a ciphertext indexed with j^*, it simply forwards it to its challenger. Otherwise, it uses the secret keys it has for all $j \neq j^*$ to answer the query.
– Eventually, it receives μ_0, μ_1 from $\mathcal{A}$, which it forwards to its challenger. Let ct^* denote the challenge ciphertext output by the challenger.
– $\mathcal{B}$ sends (j^*, ct^*) as the challenge ciphertext to $\mathcal{A}$.
– Let $\mathcal{A}$'s response be b'. $\mathcal{B}$ simply forwards b' to its challenger.

It is easy to see that $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$. Hence, $\mathcal{B}$ will win its CPA game with the same advantage as $\mathcal{A}$. □

Traceability. We prove traceability of the compiler based on the indistinguishability and adaptive (0-query) message-hiding and set-hiding properties of PSBEG below.

Theorem 21. *Assuming* PSBEG *is a secure PSBE scheme for general access structures satisfying indistinguishability, adaptive* 0*-query decoder-based set hiding, adaptive* 0*-query decoder-based message-hiding, the resulting scheme* $\mathsf{TTG}_{\mathsf{ltd}}$ *is limited traceable as per Definition 12.*

Proof (of Theorem 21). We start with introducing some notation. Given any pirate decoder box D, let $\mathcal{J}_j$ denote the value of the set $\mathcal{J}$ at the beginning of the jth iteration (or equivalently at the end of the $j-1$th iteration), i.e. $\mathcal{J}_1 = \emptyset$. By construction, $\mathcal{J}_1 \subseteq \mathcal{J}_2 \subseteq \mathcal{J}_3$ and so on. Additionally, for messages μ_0, μ_1, for any set $\mathcal{T} \subseteq [n]$ and for any $i \in \{1, \ldots, n\}$, let

$$p^D_{i,j} = \Pr\left[D(\mathsf{ct}) = b : b \leftarrow\$ \{0,1\}, \mathsf{ct} \leftarrow\$ \mathsf{EncSK}(\mathsf{msk}, \mu_b, i, \mathcal{J}_j)\right],$$

where the probability is taken over random coins of decoder D as well as the randomness used during encryption. For $i = 0$, we simply replace $\mathcal{J}_j$ by $\emptyset$:

$$p^D_{0,j} = \Pr\left[D(\mathsf{ct}) = b : b \leftarrow\$ \{0,1\}, \mathsf{ct} \leftarrow\$ \mathsf{EncSK}(\mathsf{msk}, \mu_b, 0, \emptyset)\right].$$

Similarly, let $p^D_{nrml,j} = \Pr[D((j, \mathsf{ct})) = b : b \leftarrow\$ \{0,1\}, \mathsf{ct} \leftarrow\$ \mathsf{EncPK}(\mathsf{pk}_j, \mu_b)]$.

False Trace Probability. First, we show that the tracing algorithm never falsely blames an honest user:

Theorem 22. *For every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{BadTr}] \leq \mathsf{negl}(\lambda).$$

Proof. Let J be the set of keys queried by $\mathcal{A}$. For $j \in \{1, \ldots, n\}$ and $i \in \{0, \ldots, n\}$, we define events $\mathsf{DiffAdv}^D_{i,j} : p^D_{i-1,j} - p^D_{i,j} > \epsilon/8n$ and $\mathsf{DiffAdv}^D : \bigvee_{i\in[n]\setminus J, j\in[n]} \mathsf{DiffAdv}^D_{i,j}$. Lastly, let $\mathsf{DiffAdv}^D_i : \bigvee_{j\in[n]} \mathsf{DiffAdv}^D_{i,j}$.

As in the proof of Theorem 13, we can write the probability of BadTr as a sum of conditionals:

$$\Pr[\mathsf{BadTr}] \leq \Pr[\mathsf{BadTr} | \overline{\mathsf{DiffAdv}}^D] + \Pr\left[\exists\ i \notin J \text{ s.t. } \left(\mathsf{DiffAdv}^D_i\right)\right].$$

We will now show that each of these terms is bounded by a negligible function.

Lemma 9.

$$\Pr[\mathsf{BadTr}|\ \overline{\mathsf{DiffAdv}}^{D}] \leq \mathsf{negl}(\lambda).$$

Proof. The proof of this lemma follows from Chernoff bounds. Fix any $i \in [n] \setminus J$. Let us consider the probability that the output of Trace algorithm includes i, given that $\mathsf{DiffAdv}^{D}_{i,j}$ does not occur for any j. Note that the tracing algorithm includes i in $\mathcal{J}$ if the estimates $\hat{p}_{i-1,j}$ and $\hat{p}_{i,j}$ differ by at least $\epsilon/4n$ for some $j \in [n]$.

Let $X_{a,j,\ell}$ be the random variable that is 1 if $D(\mathsf{ct}_{a,j,\ell}) = b_{a,j,\ell}$ for $a \in \{i-1, i\}$, $j \in [n]$, $\ell \in \{1, \ldots, W\}$, and let $Z_{i,j,\ell} = X_{i-1,j,\ell} - X_{i,j,\ell}$. Then, $\hat{p}_{a,j} = (\Sigma_{\ell \in [W]} X_{a,j,\ell})/W$ and $\nu_{i,j} = \mathbb{E}[Z_{i,j,\ell}] = p^{D}_{i-1,j} - p^{D}_{i,j}$.

Since the $Z_{i,j,\ell}$s are independent samples, using Chernoff bounds, we get that $\Pr[\Sigma_\ell Z_{i,j,\ell}/W > 2\nu_{i,j}] \leq 2^{-O(\lambda)}$. Using this, we get that, for every $i \in [n] \setminus J$, $\Pr[\mathsf{BadTr} \wedge i \in \mathcal{J}|\ \overline{\mathsf{DiffAdv}}^{D}] \leq n \cdot 2^{-O(\lambda)}$, by a union bound over all $j \in [n]$. By another union bound, we get that

$$\Pr[\mathsf{BadTr}|\ \overline{\mathsf{DiffAdv}}^{D}] \leq n^2 \cdot 2^{-O(\lambda)}.$$

□

Lemma 10. *If* PSBEG *is a secure PSBE scheme satisfying adaptive* 0*-bounded decoder-based set hiding, then, for all* $\lambda \in \mathbb{N}$*, for every* PPT *adversary* $\mathcal{A}$*, every non-negligible function* $\epsilon = \epsilon(\lambda)$*,*

$$\Pr\left[\exists\ i \notin J\ s.t.\ \left(\mathsf{DiffAdv}^{D}_{i}\right)\right] \leq \mathsf{negl}(\lambda)$$

Proof. Say for contradiction, the above event $\exists\ i \notin J$ s.t. $\left(\mathsf{DiffAdv}^{D}_{i}\right)$ occurs with probability $\delta(\lambda)$. We will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the decoder-based set-hiding property of PSBEG.

$\mathcal{B}$ first receives C (or n, t if threshold access structure) from $\mathcal{A}$, and forwards this to its PSBEG challenger. It then receives the PSBEG public key pk. Next, $\mathcal{B}$ chooses an index $i \leftarrow\$ \{1, \ldots, n\}$ (essentially $\mathcal{B}$ is trying to guess the honest party that $\mathcal{A}$ will try to falsely blame). Additionally, $\mathcal{B}$ chooses $j \leftarrow\$ [Y]$ – this is its guess for the iteration j in which party i will be added to the blamed set $\mathcal{J}$. $\mathcal{B}$ sets $\mathsf{pk}_j = \mathsf{pk}$, and honestly samples the remaining $Y - 1$ instances of PSBEG. Formally, for all $j' \neq j$, $\mathcal{B}$ samples $(\mathsf{pk}_{j'}, \mathsf{msk}_{j'}, (\mathsf{sk}_{j',1}, \ldots, \mathsf{sk}_{j',n})) \leftarrow\$ \mathsf{PSBEG.KeyGen}(1^\lambda, C)$. It then sends $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ to $\mathcal{A}$. Next, for every secret key query i' that $\mathcal{A}$ makes, $\mathcal{B}$ forwards it to its challenger. Let us denote the response by $\mathsf{sk}_{j,i'}$. $\mathcal{B}$ responds to $\mathcal{A}$ with $\mathsf{sk}_{1,i'}, \ldots, \mathsf{sk}_{n,i'}$ (recall it knows $\mathsf{sk}_{j',i'}$ for all $j' \neq j$). Eventually, $\mathcal{A}$ responds with a decoder box D and messages μ_0, μ_1.

$\mathcal{B}$ then runs the first $j-1$ iterations of the Trace algorithm as follows. For all $j' \in [j-1]$, it iterates over $i' = 0, \ldots, n$. For every i', it instantiates $\mathsf{count}_{i',j'} = 0$, and for each $\ell \in [W]$, it samples $b_{i',j',\ell} \leftarrow\$ \{0,1\}$. Then, if $i' = 0$, it computes $\mathsf{ct}_{i',j',\ell} = \mathsf{PSBEG.EncSK}(\mathsf{msk}_{j'}, \mu_{b_{i',j',\ell}}, i, \emptyset)$, otherwise $\mathsf{ct}_{i',j',\ell} = \mathsf{PSBEG.EncSK}(\mathsf{msk}_{j'}, \mu_{b_{i',j',\ell}}, i, \mathcal{J})$ (note $\mathcal{B}$ can do this since it knows $\mathsf{msk}_{j'}$ for all $j' \neq j$). The rest of the algorithm is the same as the Trace algorithm in Figure 10. It computes the sets $\mathcal{J}_1, \ldots, \mathcal{J}_j$ (where $\mathcal{J}_j$ is the value of the set $\mathcal{J}$ at the end of iteration $j-1$). Then, $\mathcal{B}$ samples $b' \leftarrow\$ \{0,1\}$, and outputs D, μ_b along with $(i-1, \mathcal{J}_j)$ and $(i, \mathcal{J}_j)$ if $i > 1$, otherwise it outputs $(0, \emptyset)$ and $(1, \mathcal{J}_j)$. We argue that if the $\mathsf{DiffAdv}^{D}_{i,j}$ event occurs for some $i^* \in [n], j^* \in [Y]$, and if $\mathcal{B}$ guesses these correctly, i.e. $i = i^*$ and $j = j^*$, then, $\mathcal{B}$ wins its game. To see why, observe that if $\mathcal{B}$ guesses correctly, $\mathcal{A}$ will never query the key for party i (all other key queries are allowed since for all $i' \neq i$, $i' \notin \mathcal{S}_0 \Delta \mathcal{S}_1$ as defined in the set-hiding definition). Additionally, if $\mathcal{B}$ guesses j correctly, then we

show below that the box D must be able to distinguish between encryptions of $\mu_{b'}$ to $(i-1, \mathcal{J}_j)$ vs $(i, \mathcal{J}_j)$ (and $(0, \emptyset)$ and $(1, \mathcal{J}_j)$ if $i = 1$) with non-negligible probability.

Let $p^D_{b,i'} = \Pr[D(\mathsf{ct}) = b : \mathsf{ct} \leftarrow\$ \mathsf{EncSK}(\mathsf{msk}, \mu_b, i', \mathcal{J}_j)]$, where the probability is taken over randomness of D and random coins of the encryption algorithm (for $i = 1$, we let $p^D_{b,0}$ denote the probability for ciphertext encrypted to $(0, \emptyset)$). Recall that if $\mathcal{B}$ guessed correctly, we have that i is honest and $\mathsf{DiffAdv}_{i,j}$ occurs. Therefore, we get that for i, j guessed by $\mathcal{B}$,

$$\Pr[i \notin J \wedge (p^D_{0,i-1} + p^D_{1,i-1})/2 - (p^D_{0,i} + p^D_{1,i})/2 > \epsilon/8n] \geq \delta(\lambda)/n^2.$$

This also means that there exists a bit b such that

$$\Pr[i \notin J \wedge (p^D_{b,i-1} - p^D_{b,i} \geq \epsilon/8n)] \geq \delta(\lambda)/n^2.$$

Now since $\mathcal{B}$ guesses b' uniformly at random, we have that $\mathcal{B}$ wins its game with probability $\delta/2n^2$. The lemma now follows from adaptive 0-bounded decoder-based set-hiding of PSBEG. Crucially, note that $\mathcal{B}$ never has to make an EncSK query to its challenger since it never runs the jth iteration of the Trace algorithm. □

Combining the two lemmas above, we get that the BadTr event can occur only with negligible probability. □

Correct Trace Probability. We will now show the following. First, whenever the adversary outputs a good decoder box, this box must work for ciphertexts with respect to many indices $j \in [Y]$, in particular at least t values of j (recall $t-1$ is the size of the largest unauthorized subset with respect to C). Second, for every j for which the box can decrypt, the tracer will find at least one traitor in the jth iteration. Third, we never find the same traitor for two different values of j. Combining the three, we get that the tracing algorithm will be able to find an authorized corrupt parties.

Let us start with some notation. Let $\mathsf{E}_{good,j}$ be the event that the decoder box works with probability $\epsilon' = 2/\sqrt{\lambda}$, for ciphertexts with respect to j. Formally:

$$p^D_{nrml,j} := \Pr[D((j, \mathsf{ct})) = b : b \leftarrow\$ \{0,1\}, \mathsf{ct} \leftarrow\$ \mathsf{PSBEG.EncPK}(\mathsf{pk}_j, \mu_b)] \geq \frac{1}{2} + \frac{2}{\sqrt{\lambda}}.$$

We say that D is good for index j if the event $\mathsf{E}_{good,j}$ occurs.

Next, let E_j be the event that no party was found in the jth iteration of the tracing algorithm i.e. $\mathcal{J}'_j = \emptyset$. Additionally, for $j < j'$, let $\mathsf{E}_{j,j'}$ be the event that the same party was found in iterations j and j', i.e. $\mathcal{J}'_{j'} \cap \mathcal{J}'_j \neq \emptyset$. We will now bound the probability of all these events:

Lemma 11. *In the event* $\mathsf{GoodDec}$*, D must be good for at least a $(1-\delta)$ fraction of the indices $j \in [Y]$.*

Proof. We will prove this by contradiction. Suppose this does not hold. Then, it holds that,

$$\begin{aligned}
\Pr\left[D(\mathsf{ct}) = b \ : \ b \leftarrow\!\$ \ \{0,1\} \ , \ \mathsf{ct} \leftarrow\!\$ \ \mathsf{Enc}(\mathsf{pk}, \mu_b)\right] \\
&< \delta \cdot \left(\frac{1}{2} + \frac{2}{\sqrt{\lambda}}\right) + (1-\delta) \cdot 1 \\
&= \delta \cdot \left(\frac{2}{\sqrt{\lambda}} - \frac{1}{2}\right) + 1 \\
&= \frac{\frac{1}{2} - \epsilon}{\frac{1}{2} - \frac{2}{\sqrt{\lambda}}} \cdot \left(\frac{2}{\sqrt{\lambda}} - \frac{1}{2}\right) + 1 \\
&= \frac{1}{2} + \epsilon.
\end{aligned}$$

This stands in contradiction to the assumption in the claim, and so at least $(1-\delta)$-fraction of the indices $j \in [Y]$ are good. Since we set $Y = t/(1-\delta)$, this means at least t indices in $[Y]$ are good. □

Theorem 23. *Assuming the* PSBEG *scheme achieves indistinguishability and adaptive* 0*-query decoder-based message-hiding, for every* PPT *adversary* $\mathcal{A}$ *and non-negligible function* $\epsilon(\lambda) = 2/\sqrt{\lambda}$*,*

$$\Pr[\overline{\mathsf{E}}_j] \geq \Pr[\mathsf{E}_{good,j}] - \mathsf{negl}(\lambda).$$

Proof. First, the event $\mathsf{E}_{good,j}$ implies that $p^D_{nrml,j} \geq 1/2 + \epsilon'$, where $\epsilon' = 2/\sqrt{\lambda}$. Then, by indistinguishability of PSBEG, we have that $p^D_{nrml,j} - p^D_{0,j} \leq \mathsf{negl}(\lambda)$.

Additionally, we formally prove below that $p^D_{n,j}$ must be negligible for all $j \in [Y]$, by relying on adaptive decoder-based message-hiding of PSBEG. We use $\mathsf{E}^D_{msg,j}$ to denote the event that $p^D_{n,j} > 1/2 + \delta(\lambda)$ for a non-negligible function δ, and $\mathsf{E}^D_{msg} = \bigvee_{j \in [Y]} \mathsf{E}^D_{msg,j}$.

Lemma 12. *Assuming* PSBEG *satisfies adaptive decoder-based message-hiding, for every PPT adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{E}^D_{msg}] = \mathsf{negl}(\lambda).$$

Proof. To prove this, we will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks adaptive decoder-based message-hiding of PSBEG. The game starts with $\mathcal{B}$ receiving C from $\mathcal{A}$, which it forwards to its challenger. $\mathcal{B}$ gets back $\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n$ from its challenger. $\mathcal{B}$ then guesses the index $j^* \leftarrow\!\$ \ [Y]$ for which the event $\mathsf{E}^D_{msg,j}$ occurs. It sets $\mathsf{pk}_{j^*} \leftarrow \mathsf{pk}$, and samples all other keys on its own: for all $j \neq j^*$, $(\mathsf{pk}_j, \mathsf{msk}_j, \mathsf{sk}_{j,1}, \ldots, \mathsf{sk}_{j,n}) \leftarrow\!\$ \ \mathsf{PSBEG.KeyGen}(1^\lambda, C)$. It then sends $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ to $\mathcal{A}$. Next, for any secret key query i that $\mathcal{A}$ makes, it queries its challenger for the same secret key. Let us denote the challenger's response as $\mathsf{sk}_{j^*,i}$. $\mathcal{B}$ responds to $\mathcal{A}$ with $(\mathsf{sk}_{1,i}, \ldots, \mathsf{sk}_{n,i})$. Eventually, $\mathcal{A}$ responds with a decoder D and messages μ_0, μ_1.

$\mathcal{B}$ then runs the first $j^* - 1$ iterations of the Trace algorithm, using $\mathsf{msk}_1, \ldots, \mathsf{msk}_{j^*-1}$. In particular, for iteration $j \in [j^* - 1]$, it uses msk_j to produce the EncSK ciphertexts to run the decoder with. It computes the set $\mathcal{J}_1, \ldots, \mathcal{J}_{j^*}$ as in the Trace algorithm. It then outputs $D, \mathcal{J}_{j^*}$ along with μ_0, μ_1.

We argue that if the event E^D_{msg} occurs (i.e. for some j, event $\mathsf{E}^D_{msg,j}$ occurs), and if $\mathcal{B}$ guesses j^* correctly, then $\mathcal{B}$ wins its game. To see why, by construction of the tracing algorithm, the set $\mathcal{J}_{j^*}$ must be unauthorized if $\mathcal{B}$ guessed j^* correctly (since in that case, the fact that the Trace algorithm

even executes the j^*th iteration implies that $\mathcal{J}_{j^*}$ must have been unauthorized). Hence, $\mathcal{B}$ wins the game with probability $\Pr[\mathsf{E}^D_{msg}] \cdot 1/Y$.

The theorem then follows from adaptive 0-query decoder-based message hiding of PSBEG. □

So we have, for all good $j \in [Y]$, $p^D_{nrml,j} - p^D_{0,j} \leq \mathsf{negl}(\lambda)$ i.e. $p^D_{0,j} \geq 1/2 + \epsilon(\lambda) - \mathsf{negl}(\lambda)$, and $p^D_{n,j} \leq 1/2 + \mathsf{negl}(\lambda)$. This implies $p^D_{0,j} - p^D_{n,j} \geq \epsilon - \mathsf{negl}(\lambda) \geq \epsilon/2$. Hence, there must be at least one index i^*_j such that $p^D_{i^*_j-1,j} - p^D_{i^*_j,j} > \epsilon/2n$. Then, using Chernoff bounds, we get that for all good $j \in [Y]$,

$$\Pr[\hat{p}^D_{i^*_j-1,j} - \hat{p}^D_{i^*_j,j} < \epsilon/4n] \leq 2^{-O(\lambda)} = \mathsf{negl}(\lambda).$$

This implies that for all $j \in [n]$,

$$\Pr[\overline{\mathsf{E}}_j] \geq (1 - \mathsf{negl}(\lambda)) \cdot \Pr[\mathsf{E}_{good,j}] \geq \Pr[\mathsf{E}_{good,j}] - \mathsf{negl}(\lambda).$$

Hence, if D is a good box with respect to j, we will find at least one party in the jth iteration, i.e. $\mathcal{J}'_j \neq \emptyset$ for all good $j \in [Y]$. This proves the theorem. □

To finish the proof, we will now prove that we never find the same traitor for two different values of $j \in [Y]$. Let $\mathsf{E}^D_{j,j'}$ denote the event that for some $i \in \mathcal{J}'_{j'}$, $p^D_{i-1,j} - p^D_{i,j} > \epsilon/8n$. Let $\mathsf{E}^D_{overlap}$ denote the event that for some $j < j'$, event $\mathsf{E}^D_{j,j'}$ occurs.

Lemma 13. *Assuming* PSBEG *achieves adaptive* 0*-query decoder-based set-hiding, for every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{E}^D_{overlap}] \leq \mathsf{negl}(\lambda).$$

Proof. We will construct an adversary $\mathcal{B}$ that can break set-hiding using $\mathcal{A}$, in the event $\mathsf{E}^D_{overlap}$. The game starts with $\mathcal{A}$ sending C to $\mathcal{B}$ which it forwards to its challenger. $\mathcal{B}$ gets pk from its challenger. $\mathcal{B}$ then guesses indices $j' < j \in [Y]$ for which the event $\mathsf{E}^D_{j',j}$ occurs, and guesses a party $i^* \leftarrow\!\!\$\ [n]$, such that $i^* \in \mathcal{J}'_{j'}$ and $p^D_{i^*-1,j} - p^D_{i^*,j} > \epsilon/8n$. $\mathcal{B}$ sets $\mathsf{pk}_j = \mathsf{pk}$, and samples all other public keys on its own: for all $j'' \in [Y], j'' \neq j$, $\mathcal{B}$ samples $(\mathsf{pk}_{j''}, \mathsf{msk}_{j''}, \mathsf{sk}_{j'',1}, \ldots, \mathsf{sk}_{j'',n}) \leftarrow\!\!\$\ \mathsf{PSBEG.KeyGen}(1^\lambda, C)$. $\mathcal{B}$ computes $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and sends this to $\mathcal{A}$ (note $\mathcal{B}$ also samples keys for j').

Then, for every $\mathsf{skO}(i)$ query that $\mathcal{A}$ makes, $\mathcal{B}$ forwards this query to its challenger, and let us denote the response it gets as $\mathsf{sk}_{j,i}$. $\mathcal{B}$ then responds to $\mathcal{A}$ with $(\mathsf{sk}_{1,i}, \ldots, \mathsf{sk}_{n,i})$. Eventually, $\mathcal{A}$ responds with a decoder D and messages μ_0, μ_1. $\mathcal{B}$ then runs the first $(j-1)$ iterations of the tracing algorithm: Note it does not need to query the challenger for these iterations since it knows $\mathsf{msk}_{j''}$ for all $j'' < j$, including $j'' = j'$. It computes the sets $\mathcal{J}_1 = \emptyset, \mathcal{J}_2, \ldots, \mathcal{J}_j$ as per the algorithm (recall $\mathcal{J}_j$ is the set of corrupt parties found by the end of iteration $j-1$). $\mathcal{B}$ samples $b' \leftarrow\!\!\$\ \{0,1\}$ and returns $D, \mu_{b'}$ along with $(i^*-1, \mathcal{J}_j)$, $(i^*, \mathcal{J}_j)$ to its challenger (note if $i^* = 1$, $\mathcal{B}$ will return $(0, \emptyset)$ and $(1, \mathcal{J}_j)$ but the rest of the analysis works the same way).

We argue that if the event $\mathsf{E}^D_{overlap}$ occurs and if $\mathcal{B}$ guesses j, j', i^*, b correctly, then $\mathcal{B}$ wins its set-hiding game. To see why, first observe that if $\mathcal{B}$ guesses correctly, then $i^* \in \mathcal{J}_j$ by construction (since $i^* \in \mathcal{J}'_{j'}$, which is a subset of $\mathcal{J}_j$ by construction). Additionally, $\mathcal{B}$ never makes an invalid key query, since for all $i \in [n]$, $i \in \{i^*+1, \ldots, n\} \cup \mathcal{J}_j$ if and only if $i \in \{i^*, \ldots, n\} \cup \mathcal{J}_j$ because $i^* \in \mathcal{J}_j$. Lastly, since $p^D_{i^*-1,j} - p^D_{i^*,j} > \epsilon/8n$, we can use analysis similar to Lemma 3, to argue that there exists a bit $b \in \{0,1\}$ such that the behavior of D is different for encryptions of μ_b to $(i^*-1, \mathcal{J}_j)$ vs $(i^*, \mathcal{J}_j)$.

Hence, we get that $\mathcal{B}$ wins its set hiding game with probability $\frac{1}{2n^3} \cdot \Pr[\mathsf{E}^D_{overlap}]$. Adaptive 0-query decoder based set hiding implies the theorem. □

Lemma 14.

$$\Pr[\mathsf{E}_{overlap}|\overline{\mathsf{E}^{D}_{overlap}}] \leq \mathsf{negl}(\lambda)$$

Proof. The proof directly follows the proof of Lemma 2. □

Combining all the above, we get that, for every good position $j \in [Y]$, we will find at least one unique traitor. Since there are at least t good positions, the tracing algorithm will indeed find an authorized set of traitors by the end of Y iterations.

7 Constructing PSBE for General Access Structures

In this section, we present a generic compiler to construct PSBE for general access structures from (i) a mixed FE scheme and (ii) an attribute-based encryption for the same access structure. We start with defining the two building blocks in Sections 7.1 and 7.2, before presenting the compilers in Sections 7.3 and 7.4.

7.1 Attribute-based Encryption for General Access Structures

A (key-policy) Attribute-based Encryption for general access structures (ABEG) scheme for a family of predicate classes $\mathcal{P} = \{\mathcal{P}_\lambda\}$ and a family of access structures $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ is a tuple of PPT algorithms $\mathsf{ABEG} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine})$:

- $\mathsf{Setup}(1^\lambda, C, 1^z) \to (\mathsf{pk}, \mathsf{msk})$ is the probabilistic **setup** algorithm that takes as inputs the security parameter $\lambda \in \mathbb{N}$, the access structure in the form of a circuit $C : \{0,1\}^n \to \{0,1\}$ such that $C \in \mathcal{C}_\lambda$, the attribute length parameter $z \in \mathbb{N}$, and outputs the public key pk and the master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, i, P) \to \mathsf{sk}^i_P$ is the probabilistic **key generation** algorithm that takes as inputs an index i, the master secret key msk, a predicate $P \in \mathcal{P}_\lambda$ representing a policy, and outputs a secret key sk^i_P for index i with respect to this policy. For any attribute x, we use $P(x) = 1$ to denote that x satisfies this predicate.
- $\mathsf{Enc}(\mathsf{pk}, \mu, x) \to \mathsf{ct}_x$ is the probabilistic **encryption** algorithm that takes as inputs the public key pk, the plain text message μ, an attribute x and outputs a ciphertext ct_x. We may use $\mathsf{Enc}(\mathsf{pk}, \mu, x; r)$ to denote the deterministic version of this algorithm, which takes as input randomness r sampled from a randomness space $\mathcal{R}$.
- $\mathsf{Dec}(\mathsf{sk}^i_P, \mathsf{ct}_x) \to d_i$ is the probabilistic **decryption** algorithm that takes as input a secret key sk^i_P for some index i and policy P, a ciphertext ct_x generated with respect to attribute x and outputs the partial decryption d_i of ct_x under sk^i_P.
- $\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}_x, \{d_j\}_{j\in\mathcal{S}}) \to \mu$ is the deterministic **combiner** algorithm that takes as inputs a set of decryption shares $\{d_j\}_{j\in\mathcal{S}}$ for a set $\mathcal{S} \subseteq [n]$ of size t and outputs a message μ.

Correctness. An ABEG scheme ABEG must satisfy correctness, which requires that any subset of parties that satisfies the access structure must be able to decrypt an honestly generated ciphertext for any attribute, as long as the attribute satisfies the policy associated with the keys of *all* parties in the subset. Definition 22 formalizes this notion.

Definition 22 (Correctness). *An Attribute-based encryption for general access structures scheme* $\mathsf{ABEG} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine})$ *is correct if, for all* $\lambda \in \mathbb{N}$*, all* $C \in \mathcal{C}_\lambda$*, all* $z \in \mathbb{N}$*, all attributes* x *in the attribute space, all messages* μ*, all subsets* $\mathcal{S}$ *such that* $C(x_\mathcal{S}) = 1$*, and corresponding policies* $\{P_i\}_{i\in\mathcal{S}}$ *such that* x *satisfies* P_i *for all* $i \in \mathcal{S}$*, the following probability is negligible in* λ*:*

$$\Pr\left[\mu' \neq \mu \;:\; \begin{array}{c} (\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\; \mathsf{Setup}(1^\lambda, C, 1^z) \\ \mathsf{sk}^i_{P_i} \leftarrow\!\$\; \mathsf{KeyGen}(\mathsf{msk}, i, P_i)\ \forall\ i \in \mathcal{S} \\ \mathsf{ct}_x \leftarrow\!\$\; \mathsf{Enc}(\mathsf{pk}, \mu, x) \\ d_i := \mathsf{Dec}(\mathsf{sk}^i_{P_i}, \mathsf{ct}_x)\ \forall\ i \in \mathcal{S} \\ \mu' := \mathsf{Combine}(\mathsf{pk}, \mathsf{ct}_x, \{d_j\}_{j\in\mathcal{S}}) \end{array}\right]$$

Semantic Security. We define semantic security for ABEG by generalizing the definition considered in [DDP+25] for attribute-based threshold encryption to general access structures. Roughly speaking, we require that an adversary cannot learn anything about the encrypted message for an attribute, even if it is allowed to (i) query arbitrarily many keys for policies not satisfied by the attribute (as in ABE security), (ii) query keys for an unauthorized set of parties with policies satisfied by the attribute (as in threshold encryption security). We also allow the adversary to query decryption shares for any *valid* ciphertext.

In the formal definition below, we define adaptive security, wherein the adversary adaptively chooses which parties to corrupt, and can choose the challenge attribute x^* at any point in the game. But we also consider selective adversary, wherein the adversary specifies x^* in the beginning of the game, and makes all the key queries for satisfying policies before making other queries.

Definition 23 (CPA Security). *We say that* ABEG *satisfies (adaptive) selective CPA security if for every probabilistic polynomial time adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{x-cpa}}_{\mathcal{A},\mathsf{ABEG}}(\lambda) := \left|\frac{1}{2} - \Pr[\mathbf{IND\text{-}CPA}^{x}_{\mathcal{A},\mathsf{ABEG}}(\lambda) = 1]\right|,$$

where $x \in \{sel, adap\}$*.*

7.2 Mixed Functional Encryption

Consider function classes $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ and message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$, where $f : \mathcal{M}_\lambda \to \{0,1\}$ for each $f \in \mathcal{F}_\lambda$. A mixed functional encryption scheme Mixed-FE [GKW18], for function classes $\mathcal{F}$ and message spaces $\mathcal{M}$, consists of five PPT algorithms $\mathsf{MFE} = (\mathsf{Setup}, \mathsf{EncSK}, \mathsf{EncPK}, \mathsf{KeyGen}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ, and outputs the public key pk and the master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mu) \to \mathsf{sk}_\mu$. The key generation algorithm takes as input master secret key msk and a message $\mu \in \mathcal{M}_\lambda$. It outputs a secret key sk_μ.
- $\mathsf{EncPK}(\mathsf{pk}) \to \mathsf{ct}$. The normal encryption algorithm takes as input the public key pk, and outputs a ciphertext ct.
- $\mathsf{EncSK}(\mathsf{msk}, f) \to \mathsf{ct}$. The secret key encryption algorithm takes as input master secret key msk and a function $f \in \mathcal{F}_\lambda$. It outputs a ciphertext ct.

Experiment $\mathbf{IND\text{-}CPA}^{adap}_{\mathcal{A},\mathsf{ABEG}}(\lambda)$

$\mathcal{J} := \emptyset\ ;\ x^* \leftarrow \perp\ ;\ M := \emptyset$

$(C, z, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$

$(\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, C, 1^z)$

$(\mu_0, \mu_1, x^*, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}^{\mathsf{sk}(\cdot),\mathsf{dec}(\cdot)}(\mathsf{state}, \mathsf{pk})$

$b \leftarrow\!\$\ \{0,1\},\ \mathsf{ct}^* \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mu_b, x^*)$

$b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{sk}(\cdot),\mathsf{dec}(\cdot)}(\mathsf{state}, \mathsf{ct}^*)$

$\mathcal{J} = \{i \in [n] : \exists\ (i, P_i) \in M \text{ s.t. } P_i(x^*) = 1\}$

$\mathcal{J}_2 = \{i \in [n] : \exists\ (i, P_{i,1}) \in M \wedge (i, P_{i,2}) \in M \text{ s.t. } P_{i,1}(x^*) = 1 \wedge P_{i_2}(x^*) = 1 \wedge P_{i,1} \neq P_{i,2}\}$

if $C(x_{\mathcal{J}_1}) = 1 \vee \mathcal{J}_2 \neq \emptyset$ **then return** 0

return $b' = b$

Oracle $\mathsf{sk}(i, P_i)$

if $i \notin [n]$: **return** $\perp$

$M \leftarrow M \cup \{i, P_i\}$

return $\mathsf{sk}^i_{P_i} \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{msk}, i, P_i)$

Oracle $\mathsf{dec}(\mu, r, x, i, P_i)$

if $i \notin [n] \vee P_i(x) = 0$:

return $\perp$

$\mathsf{ct} := \mathsf{Enc}(\mathsf{pk}, \mu, x; r)$

$\mathsf{sk}^i_{P_i} \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{msk}, i, P_i)$

return $\mathsf{Dec}(\mathsf{sk}^i_{P_i}, \mathsf{ct})$

Fig. 11. The adaptive IND-CPA security experiment for an attribute-based encryption scheme for general access structures ABEG and an adversary $\mathcal{A}$.

– $\mathsf{Dec}(\mathsf{sk}_\mu, \mathsf{ct}) \to \{0,1\}$. The decryption algorithm takes as input a secret key sk_μ and a ciphertext ct, and it outputs a single bit.

Definition 24 (Correctness). *A mixed functional encryption scheme* MFE *is said to satisfy correctness if there exists negligible functions* negl_1, negl_2 *such that for all* $\lambda \in \mathbb{N}$, *and for all messages* $\mu \in \mathcal{M}_\lambda$ *and all functions* $f \in \mathcal{F}_\lambda$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_\mu, \mathsf{ct}) = 1 \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_\mu \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{msk}, \mu) \\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncPK}(\mathsf{pk})\end{array}\right] \geq 1 - \mathsf{negl}_1(\lambda)$$

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_\mu, \mathsf{ct}) = f(m) \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_\mu \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{msk}, \mu) \\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, f)\end{array}\right] \geq 1 - \mathsf{negl}_2(\lambda)$$

q-Bounded Security. The security of an MFE scheme is captured through the notions of *function indistinguishability* and *accept indistinguishability* in the presence of an adversary that makes at most q many secret key encryption queries and unbounded number of key generation queries. The first property states that for any two functions $f_0, f_1 \in \mathcal{F}_\lambda$, if the output of the two functions is identical on every μ such that sk_μ is in the adversary's possession, then any PPT adversary cannot distinguish between the secret key encryptions of f_0 and f_1. The *accept indistinguishability* property states that the secret key encryption of a function $f \in \mathcal{F}_\lambda$ and a public key encryption should look

indistinguishable for any PPT adversary that only possesses keys for messages μ such that $f(\mu) = 1$. We define both the adaptive and selective versions of these two properties below.

Definition 25 (q-bounded Function Indistinguishability). *Let $q(\cdot)$ be any fixed polynomial. A Mixed-FE scheme* MFE *is said to satisfy q-bounded function indistinguishability if there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$, and for any* PPT *adversary $\mathcal{A}$,*

$$\Pr\left[b' = b \ : \begin{array}{c} (\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ f_0, f_1 \leftarrow\!\$\ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{EncSK}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{pk}) \\ b \leftarrow\!\$\ \{0,1\} \\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, f_b) \\ b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{EncSK}(\mathsf{msk},\cdot)}(\mathsf{ct}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where

- *$\mathcal{A}$ can make at most $q(\lambda)$ many queries to* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$ *and,*
- *f_0 and f_1 agree on every input μ submitted by $\mathcal{A}$ to* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$*, i.e. $f_0(\mu) = f_1(\mu)$.*

We also consider the selective version of function indistinguishability as defined in [DDP+25, GKW18]. Herein, the adversary needs to declare the two challenge functions $f_0, f_1 \in \mathcal{F}_\lambda$ at the beginning of the security game and must submit all its q encryption queries before submitting any key generation query.

Definition 26 (q-bounded Selective Function Indistinguishability). *Let $q(\cdot)$ be any fixed polynomial. A Mixed-FE scheme* MFE *is said to satisfy q-bounded selective function indistinguishability if there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$, and for any* PPT *adversary $\mathcal{A}$,*

$$\Pr\left[b' = b \ : \begin{array}{c} f_0, f_1 \leftarrow\!\$\ \mathcal{A}(1^\lambda) \\ (\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ b \leftarrow\!\$\ \{0,1\} \\ \mathsf{ct} \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, f_b) \\ b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{EncSK}(\mathsf{msk},\cdot)}(\mathsf{pk}, \mathsf{ct}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where

- *$\mathcal{A}$ can make at most $q(\lambda)$ many queries to* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$ *and,*
- *f_0 and f_1 agree on every input μ submitted by $\mathcal{A}$ to* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$*, i.e. $f_0(\mu) = f_1(\mu)$, and*
- *$\mathcal{A}$ must make all (at most $q(\lambda)$)* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$ *oracle queries before making any query to* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *oracle.*

We now define the adaptive and selective notions of accept indistinguishability:

Definition 27 (q-bounded Accept Indistinguishability). *A Mixed-FE scheme* MFE *is said to satisfy q-bounded accept indistinguishability if there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$, and for any* PPT *adversary $\mathcal{A}$,*

$$\Pr\left[b' = b \ : \begin{array}{c} (\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ f^* \leftarrow\!\$\ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{EncSK}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{pk}) \\ b \leftarrow\!\$\ \{0,1\} \\ \mathsf{ct}_0 \leftarrow\!\$\ \mathsf{EncPK}(\mathsf{pk})\ ;\ \mathsf{ct}_1 \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, f^*) \\ b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{EncSK}(\mathsf{msk},\cdot)}(\mathsf{ct}_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where

– $\mathcal{A}$ *can make at most* $q(\lambda)$ *many queries to* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$ *and,*
– *every query* μ *that* $\mathcal{A}$ *submits to* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *should satisfy* $f^*(\mu) = 1$ *as well as* $f(\mu) = 1$ *for every query* f *that* $\mathcal{A}$ *makes to* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$.

Definition 28 (q-bounded Selective Accept Indistinguishability). *A Mixed-FE scheme* MFE *is said to satisfy* q*-bounded selective accept indistinguishability if there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*, and for any* PPT *adversary* $\mathcal{A}$*,*

$$\Pr\left[b' = b \ : \begin{array}{c} f^* \leftarrow\!\$\ \mathcal{A}(1^\lambda) \\ (\mathsf{pk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ b \leftarrow\!\$\ \{0,1\} \\ \mathsf{ct}_0 \leftarrow\!\$\ \mathsf{EncPK}(\mathsf{pk})\ ;\ \mathsf{ct}_1 \leftarrow\!\$\ \mathsf{EncSK}(\mathsf{msk}, f^*) \\ b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{EncSK}(\mathsf{msk},\cdot)}(\mathsf{pk}, \mathsf{ct}_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where

– $\mathcal{A}$ *can make at most* $q(\lambda)$ *many queries to* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$ *and,*
– *every query* μ *that* $\mathcal{A}$ *submits to* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *should satisfy* $f^*(\mu) = 1$ *as well as* $f(\mu) = 1$ *for every query* f *that* $\mathcal{A}$ *makes to* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$
– $\mathcal{A}$ *must make all (at most* $q(\lambda)$*)* $\mathsf{EncSK}(\mathsf{msk}, \cdot)$ *oracle queries before making any query to* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *oracle.*

7.3 PSBE for General Access Structures from ABEG and MFE

We start with describing the rough idea below before presenting the formal compiler in Figure 12: works as follows:

– The ciphertext is an ABEG ciphertext with an attribute which is either a 'normal' MFE ciphertext encrypting the identity function or a 'special' MFE ciphertext encrypting a special comparison function. In more detail, the ciphertexts generated in the public-key mode (via MFE.EncPK) have as attribute the normal MFE ciphertext. For encrypting a message in the secret-key mode with respect to an index j and a set $\mathcal{T}$, the attribute is the MFE ciphertext encrypting the special comparison function $f_{>j,\in\mathcal{T}}$; for any input i, this function outputs 1 only if either $i > j$ or $i \in \mathcal{T}$.
– The secret key of each party i is an ABEG secret key, with the policy as the MFE decryption circuit hard-coded with the i-th MFE secret key.
– To decrypt, if the ciphertext attribute is a normal MFE ciphertext then each party can use their ABEG secret key to partially decrypt the ABEG ciphertext. Finally, partial decryptions of an authorized set of parties can be combined to recover the message. However, if the ciphertext attribute is a 'special' MFE ciphertext, only users with a policy $(\mathsf{MFE.Dec}(\mathsf{MFE.sk}_i, \cdot))$ that is satisfied by the ciphertext attribute are able to partially decrypt. This means that, a ciphertext encrypted with respect to index j and set $\mathcal{T}$ can only be decrypted by a set of authorized parties all with index $i > j \vee i \in \mathcal{T}$.

Theorem 24. *If the underlying* MFE *and* ABEG *schemes are correct as per definitions Def. 24 and Def. 22 resp., then the resulting* PSBEG *scheme also satisfies correctness.*

Proof. We show that our PSBEG scheme is correct both in the public key and secret key modes.

$\underline{\mathsf{KeyGen}(1^\lambda, C) \to (\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) :}$

1. Sample $(\mathsf{MFE.pk}, \mathsf{MFE.msk}) \leftarrow\!\$\ \mathsf{MFE.Setup}(1^\lambda)$.
2. Sample the MFE secret key for user i as $\mathsf{MFE.sk}_i \leftarrow\!\$\ \mathsf{MFE.KeyGen}(\mathsf{MFE.msk}, i)$.
3. Set $P_i := \mathsf{MFE.Dec}(\mathsf{MFE.sk}_i, \cdot)$.
4. Sample $(\mathsf{ABEG.pk}, \mathsf{ABEG.msk}) \leftarrow\!\$\ \mathsf{ABEG.Setup}(1^\lambda, C, 1^z)$. // z is the ciphertext length in MFE.
5. Sample $\mathsf{sk}_i \leftarrow\!\$\ \mathsf{ABEG.KeyGen}(\mathsf{ABEG.msk}, i, P_i)\ \forall\ i\ \in [n]$.
6. Set $\mathsf{pk} := (\mathsf{MFE.pk}, \mathsf{ABEG.pk})$ and $\mathsf{msk} = \mathsf{MFE.msk}$.
7. Output the tuple $(\mathsf{pk}, \mathsf{msk})$ and the secret keys $\{\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n\}$.

$\underline{\mathsf{EncPK}(\mathsf{pk}, \mu) :}$

1. Parse pk as $(\mathsf{MFE.pk}, \mathsf{ABEG.pk})$.
2. Compute $x \leftarrow\!\$\ \mathsf{MFE.EncPK}(\mathsf{MFE.pk})$.
3. Output $\mathsf{ct}_x \leftarrow\!\$\ \mathsf{ABEG.Enc}(\mathsf{ABEG.pk}, \mu, x)$

$\underline{\mathsf{EncSK}(\mathsf{msk}, \mu, j, \mathcal{T}) :}$

1. Parse msk as MFE.msk.
2. Compute $x \leftarrow\!\$\ \mathsf{MFE.EncSK}(\mathsf{MFE.msk}, \mathsf{comp}_{j,\mathcal{T}})$, where $\mathsf{comp}_{j,\mathcal{T}} : [n] \to \{0,1\}$, such that $\mathsf{comp}_{j,\mathcal{T}}(i) = 1\ \forall\ i > j\ \wedge \forall\ i \in \mathcal{T}$ and 0 otherwise.
3. Output $\mathsf{ct}_x \leftarrow\!\$\ \mathsf{ABEG.Enc}(\mathsf{ABEG.pk}, \mu, x)$.

$\underline{\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_x) :}$

1. Output $d_i \leftarrow \mathsf{ABEG.Dec}(\mathsf{sk}_i, \mathsf{ct}_x)$.

$\underline{\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}_x, \{d_j\}_{j\in\mathcal{S}}) :}$

1. Parse pk as $(\mathsf{MFE.pk}, \mathsf{ABEG.pk})$.
2. Output $\mathsf{ABEG.Combine}(\mathsf{ABEG.pk}, \mathsf{ct}_x, \{d_j\}_{j\in\mathcal{S}})$.

Fig. 12. Our compiler to construct PSBEG from Mixed functional encryption MFE and attribute-based encryption for general access structures ABEG.

– In the public-key encryption mode, the ciphertext attribute x is a normal MFE ciphertext, computed using the public key mode of MFE as $x \leftarrow\!\$\ \mathsf{MFE.EncPK}(\mathsf{MFE.pk})$. The function f that is encrypted in a normal MFE ciphertext is the always-accepting function. So from correctness of MFE, it follows that $\mathsf{MFE.Dec}(\mathsf{MFE.sk}_i, x) = f(i) = 1$ for all $i \in [n]$ except with negligible probability. This implies that the ciphertext attribute x satisfies the key-policy associated with $\mathsf{ABEG.sk}_i$ for all $i \in [n]$. Therefore, each party is able to decrypt ct_x using ABEG decryption under their key $\mathsf{ABEG.sk}_i$ to get $d_i \leftarrow \mathsf{ABEG.Dec}(\mathsf{ABEG.sk}_i, \mathsf{ct}_x)$ and by correctness of ABEG, partial decryptions of a set S of authorized parties (i.e. $C(x_S) = 1$) can be combined to recover the message i.e. $\mathsf{ABEG.Combine}(\mathsf{ct}_x, \{d_i\}_{i\in S}) = m$.

– The ciphertext attribute x in the secret-key encryption mode is a MFE ciphertext generated as $x \leftarrow\!\$\ \mathsf{MFE.Enc}(\mathsf{MFE.msk}, f_{>j,\in\mathcal{T}})$ for some index j and some set $\mathcal{T}$, where $f_{>j,\in\mathcal{T}}$ is the special comparison function that outputs 1 on all inputs $i > j$, $i \in \mathcal{T}$, and 0 otherwise. From correctness of the MFE scheme, $\mathsf{MFE.Dec}(\mathsf{MFE.sk}_i, x) = 1$ iff $i > j \vee i \in \mathcal{T}$ and 0 for $i \leq j \wedge i \notin \mathcal{T}$. So the ciphertext attribute x satisfies the policy associated with $\mathsf{ABEG.sk}_i$ only for indices $i > j \vee i \in \mathcal{T}$. Consequently, only parties with indices $i > j$ or $i \in \mathcal{T}$ are able to decrypt ct_x using ABEG decryption. By correctness of ABEG, for any set S authorized with respect to C,

partial decryptions d_i for all parties $i \in S$ with $i > j \vee i \in \mathcal{T}$ can be combined to recover the message, i.e. $\mathsf{ABEG.Combine}(\mathsf{ct}_x, \{d_i\}_{i\in S}) = \mu$.

□

Theorem 25. *If the* ABEG *scheme satisfies IND-CPA security, the resulting* PSBEG *scheme also satisfies IND-CPA security.*

Proof. The proof essentially follows the proof of Theorem 12 in [DDP+25]: Using an adversary $\mathcal{A}$ that breaks IND-CPA security of PSBEG, we can construct an adversary $\mathcal{B}$ that breaks IND-CPA of ABEG: $\mathcal{B}$ runs $\mathsf{MFE.Setup}(1^\lambda)$ and sets the challenge attribute to be a normal MFE ciphertext, that is, $x^* = \mathsf{MFE.EncPK}(\mathsf{MFE.pk})$. It then forwards C along with x^* to its challenger, and relays the response to $\mathcal{A}$. For any secret key query made by $\mathcal{A}$, $\mathcal{B}$ queries its challenger for the secret key for (i, P_i) wherein P_i is defined exactly as in the scheme. $\mathcal{B}$ simply forwards the challenger's response to $\mathcal{A}$.

For decryption oracle queries, $\mathcal{B}$ parses the ciphertext randomness r as (r_0, r_1) and queries its challenger on $\mathsf{dec}(\mu, r_1, x, P_i)$ where P_i is the same as above, and $x = \mathsf{MFE.EncPK}(\mathsf{MFE.pk}; r_0)$. For the challenge messages and the challenge ciphertexts, $\mathcal{B}$ simply acts as a relay between the challenger and $\mathcal{A}$.

It is easy to see that $\mathcal{B}$ perfectly simulates the IND-CPA game to $\mathcal{A}$. Additionally, $\mathcal{B}$ wins its IND-CPA game with the same probability as $\mathcal{A}$. □

Theorem 26. *The* PSBEG *scheme satisfies q-bounded indistinguishability as long as the underlying* MFE *scheme is q-bounded restricted accept indistinguishable.*

Proof. The proof of this theorem is a straightforward generalization of the proof of [DDP+25, Theorem 13] to support general access structures. In more detail, we construct an adversary $\mathcal{B}$ for q-bounded restricted accept indistinguishability, that uses an adversary $\mathcal{A}$ for q-bounded indistinguishability. It receives parameters C from $\mathcal{A}$, It sends the function $\mathsf{comp}_{0,\emptyset}$ as its challenge function to its challenger, and receives MFE parameters along with the challenge ciphertext ct^* in return. $\mathcal{B}$ samples the ABEG parameters for C on its own. It then queries its challenger for q encryptions to the function $\mathsf{comp}_{0,\emptyset}$, which it can use to respond to $\mathsf{EncSK}(\cdot, 0, \emptyset)$ queries by $\mathcal{A}$. The rest of the reduction is the same as that for the proof of [GKW18, Lemma 6.1]

At a high level, the reason why the access structure does not affect the proof is that, in the security argument, the adversary is allowed to learn all n ABEG decryption keys, meaning that it is capable of decrypting the challenge ciphertext. Hence, the access structure does not play any role here, and we get indistiguishability simply owing to the q-bounded accept indistinguishability of the MFE. □

Theorem 27. *Our* PSBEG *scheme satisfies selective (adaptive) q-bounded message hiding as long as the underlying* ABEG *achieves selective (adaptive) CPA security.*

Proof. The proof for the selective case follows the proof of Theorem 15 in [DDP+25]: We construct $\mathcal{B}$ to break selective CPA security of ABEG using any adversary $\mathcal{A}$ that breaks selective q-bounded message hiding. $\mathcal{B}$ receives $C, \mathcal{J}$ from $\mathcal{A}$. It runs MFE.Setup and produces the challenge attribute as the secret-key MFE ciphertext encrypting $\mathsf{comp}_{n,\mathcal{J}}$, i.e. $x^* \leftarrow\!\!\$\ \mathsf{MFE.EncSK}(\mathsf{MFE.msk}, \mathsf{comp}_{n,\mathcal{J}})$. $\mathcal{B}$ sends (C, x^*) to its challenger, and then forwards the ABEG public key to $\mathcal{A}$. $\mathcal{B}$ queries the challenger for secret keys for (i, P_i) for all $i \in [n]$, where P_i is as defined in the scheme: note $\mathcal{B}$ can

do this since only parties in $\mathcal{J}$ have keys for satisfying policies, and $\mathcal{J}$ must be unauthorized for $\mathcal{A}$ to win. Hence, if $\mathcal{A}$ wins, then $\mathcal{B}$ only queries an unauthorized set of keys with policies satisfied by x^*. $\mathcal{B}$ then forwards all these keys to $\mathcal{A}$. Lastly, $\mathcal{B}$ can respond to any number of secret-key encryption queries since it has $\mathsf{MFE.msk}$. In the challenge phase, $\mathcal{B}$ simply acts as a relay between the challenger and $\mathcal{A}$.

It is easy to see that $\mathcal{B}$ perfectly simulates the message-hiding game to $\mathcal{A}$. Moreover, $\mathcal{B}$ wins its game with the same advantage as $\mathcal{A}$.

The proof for the adaptive case is the same, except that $\mathcal{A}$ sends $\mathcal{J}$ during the challenge phase, and so $\mathcal{B}$ generates x^* (as above) and sends it to its challenger along with messages μ_0, μ_1 (instead of in the beginning of the game). □

Theorem 28. *The* PSBEG *scheme satisfies selective (adaptive) q-bounded set hiding as long as the underlying* MFE *scheme is q-bounded selective (adaptive) function indistinguishable.*

Proof. Starting with the selective version: the game starts with $\mathcal{B}$ receiving $C, (i_0, \mathcal{T}_0), (i_1, \mathcal{T}_1)$ from $\mathcal{A}$. It forwards $f_b := \mathsf{comp}_{i_b, \mathcal{T}_b}$ for $b \in \{0, 1\}$ as the challenge functions to its challenger. $\mathcal{B}$ receives the pk, ct from its challenger. It then samples $(\mathsf{ABEG.pk}, \mathsf{ABEG.msk}) \leftarrow\!\!\$\ \mathsf{ABEG.Setup}(1^\lambda, C, z)$, and sends the overall public key $(\mathsf{ABEG.pk}, \mathsf{pk})$ to $\mathcal{A}$. For any $\mathsf{EncSK}(\mu, i, \mathcal{T})$ query, it queries its challenger for $\mathsf{EncSK}(\mathsf{comp}_{i,\mathcal{T}})$, let the response be x. It then responds to $\mathcal{A}$ with $\mathsf{ABEG.Enc}(\mathsf{ABEG.pk}, \mu, x)$. For any $\mathsf{skO}(i)$ query made by $\mathcal{A}$, $\mathcal{B}$ queries its challenger for secret key of i, let the response be sk_i. $\mathcal{B}$ then responds with $\mathsf{ABEG.KeyGen}(\mathsf{ABEG.msk}, i, P_i)$ where P_i is the MFE decryption circuit with sk_i hard-coded. Eventually, $\mathcal{A}$ outputs a message μ, and $\mathcal{B}$ gets the challenge ciphertext ct^* from its challenger. $\mathcal{B}$ sends to $\mathcal{A}$ $\mathsf{ABEG.Enc}(\mathsf{pk}, \mu, \mathsf{ct}^*)$. Then, $\mathcal{B}$ simply forwards $\mathcal{A}$'s response b' to its challenger.

Clearly, $\mathcal{B}$ perfectly simulates the set hiding game to $\mathcal{A}$. Additionally, it is an admissible adversary for the function indistinguishability game if $\mathcal{A}$ wins its game. To see why, first observe that $\mathcal{A}$ must make all EncSK queries before querying any secret keys – hence $\mathcal{B}$ also does the same. Next, $\mathcal{A}$ is only allowed to query keys for parties outside $\mathcal{S}_0 \Delta \mathcal{S}_1$: implying that for any query $\mathcal{B}$ makes, f_0 and f_1 will agree on that query. Lastly, $\mathcal{A}$ can only make q EncSK queries. Hence, $\mathcal{B}$ wins its game with the same advantage as $\mathcal{A}$.

For the adaptive case: the game starts with $\mathcal{B}$ receiving C from $\mathcal{A}$, and receiving pk from its challenger. $\mathcal{B}$ samples $(\mathsf{ABEG.pk}, \mathsf{ABEG.msk}) \leftarrow\!\!\$\ \mathsf{ABEG.Setup}(1^\lambda, C, z)$, and sends the overall public key $(\mathsf{ABEG.pk}, \mathsf{pk})$ to $\mathcal{A}$. $\mathcal{B}$ handles oracle queries made by $\mathcal{A}$ in the same way as in the selective game. Eventually, it receives $(\mu, i_0, \mathcal{T}_0, i_1, \mathcal{T}_1)$ from $\mathcal{A}$, then $\mathcal{B}$ sends f_0, f_1 exactly as defined above to its challenger. Upon receiving ct^* from its challenger, $\mathcal{B}$ sends to $\mathcal{A}$ $\mathsf{ABEG.Enc}(\mathsf{pk}, \mu, \mathsf{ct}^*)$. Then, $\mathcal{B}$ simply forwards $\mathcal{A}$'s response b' to its challenger. By similar arguments as in the selective case, $\mathcal{B}$ is an admissible adversary and wins its game with the same advantage as that of $\mathcal{A}$. □

7.4 q-bounded Decoder-based PSBEG from $q+1$-bounded PSBEG

Let PSBEG be a PSBE scheme for general access structures that satisfies adaptive $q+1$-bounded security. Then, we prove that the same scheme also satisfies adaptive q-query decoder-based security. We focus on adaptive security but the same argument applies for selective security definitions as well.

Lemma 15. *If* PSBEG *satisfies 1-bounded indistinguishability (Definition 15), then it also satisfies decoder-based indistinguishability (Definition 19) property.*

Proof. Same as proof of Lemma 4.1 of [GKW18]. □

Lemma 16. *If* PSBEG *satisfies adaptive $q+1$-bounded message hiding (Definition 16), then it also satisfies adaptive q-query decoder-based message-hiding (Definition 21) property.*

Proof. The proof essentially follows the proof of [GKW18, Lemma 4.3]. In particular, given an adversary $\mathcal{A}$ that breaks (adaptive) q-query decoder-based message-hiding, we will use it to construct $\mathcal{B}$ that breaks (adaptive) $q+1$-bounded message hiding. The game starts with $\mathcal{B}$ receiving C from $\mathcal{A}$, which it forwards to its challenger. $\mathcal{B}$ gets back $\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n$, which it forwards to $\mathcal{A}$. Additionally, $\mathcal{B}$ simply forwards any EncSK queries that $\mathcal{A}$ makes, to its challenger, and relays the response. Eventually, $\mathcal{A}$ outputs $D, \mu_0, \mu_1, \mathcal{J}$. $\mathcal{B}$ samples $\beta \leftarrow\!\!\$\ \{0,1\}$, and queries the challenger for $\mathsf{EncSK}(\mu_b, n, \mathcal{J})$ (note that $\mathcal{B}$ is allowed one more query than $\mathcal{A}$), let the response be ct_β. $\mathcal{B}$ then sends $(\mu_0, \mu_1, \mathcal{J})$ to its challenger, and it gets back the challenge ciphertext ct^*. $\mathcal{B}$ outputs $b' = \beta$ if $D(\mathsf{ct}^*) = D(\mathsf{ct}_\beta)$, and $b' = 1 - \beta$ otherwise.

We can use the same argument as that for [GKW18, Lemma 4.3] to argue that $\mathcal{B}$ does indeed win its game with non-negligible advantage if $\mathcal{A}$ wins its game. □

Lemma 17. *If* PSBEG *satisfies (adaptive) $q+1$-bounded set hiding (Definition 17), then it also satisfies (adaptive) q-query decoder-based set-hiding (Definition 20) property.*

Proof. The proof follows the proof of Lemma 16. Given an adversary $\mathcal{A}$ that breaks (adaptive) q-query decoder-based set-hiding, we will use it to construct $\mathcal{B}$ that breaks (adaptive) $q+1$-bounded set hiding. The game starts with $\mathcal{B}$ receiving C from $\mathcal{A}$, which it forwards to its challenger. $\mathcal{B}$ gets back pk, which it forwards to $\mathcal{A}$. Next, for any EncSKO or skO query made by $\mathcal{A}$, it simply forwards it to its challenger and relays the response back to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs $D, \mu, i_0, \mathcal{J}_0, i_1, \mathcal{J}_1$. $\mathcal{B}$ samples $\beta \leftarrow\!\!\$\ \{0,1\}$ and queries its challenger for $\mathsf{EncSK}(\mu, i_\beta, \mathcal{J}_\beta)$ (note $\mathcal{B}$ is allowed to make one more query in its game than $\mathcal{A}$), let the response be ct_β. $\mathcal{B}$ then sends $\mu, i_0, \mathcal{J}_0, i_1, \mathcal{J}_1$ to its challenger, and receives the challenge ciphertext ct^*. $\mathcal{B}$ outputs $b' = \beta$ if $D(\mathsf{ct}^*) = D(\mathsf{ct}_\beta)$, and $b' = 1 - \beta$ otherwise.

By the same argument as in proof of [GKW18, Lemma 4.3] we get that $\mathcal{B}$ does indeed win its game with non-negligible advantage if $\mathcal{A}$ wins its game. □

8 Risky and Limited Traitor Tracing from Weak PSBE

In this section, we present our generic compiler to construct a Risky and Limited TTG scheme from a weak PSBEG. We start with defining weak PSBEGen, before describing our compiler.

Definition 29 (Weak Decoder-based Indistinguishability). *A* PSBEG *scheme is said to satisfy weak selective decoder-based indistinguishability if there are constants $\alpha, \beta, \gamma \in (0,1)$ such that, for every polynomial $\kappa(\lambda)$ and every stateful* PPT *adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for every $\lambda \in \mathbb{N}$, the following is true:*

- *Run $\mathcal{A}(1^\lambda)$ to get C.*
- *Now run $(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, C)$ and send $\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}$ to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a decoder D and messages μ_0, μ_1.*

– *Let* $\mathsf{Adv}(D,\mathsf{pk}) = 2\Pr[D(\mathsf{ct}) = b : \substack{b\leftarrow\$\{0,1\} \\ \mathsf{ct}\leftarrow\$\mathsf{EncPK}(\mathsf{pk},\mu_b)}] - 1$ *be the advantage of* D *in decrypting for publicly generated ciphertexts. Let* $\mathsf{Adv}'(D,\mathsf{msk}) = 2\Pr[D(\mathsf{ct}) = b : \substack{b\leftarrow\$\{0,1\} \\ \mathsf{ct}\leftarrow\$\mathsf{EncSK}(\mathsf{msk},\mu_b,0,\emptyset)}] - 1$ *be the advantage for secret-key ciphertexts for index* 0 *and set* $\emptyset$*.*
Let $\mathsf{Good}_\alpha(D,\mathsf{pk})$ *be the event* $\mathsf{Adv}(D,\mathsf{pk}) \geq \alpha$*, and let* $\mathsf{Good}'_\beta(D,\mathsf{msk})$ *be the event* $\mathsf{Adv}'(D,\mathsf{msk}) \geq \beta$*. Then we have* $\Pr[\mathsf{Good}'_\beta(D,\mathsf{msk})] \geq \gamma \Pr[\mathsf{Good}_\alpha(D,\mathsf{pk})] - \mathsf{negl}(\lambda)$.

Definition 30 (Weak PSBE for General Access Structures). *A* PSBEG *scheme is said to be a* (α,β,γ)*-weak PSBEGen if it satisfies* (α,β,γ)*-weak Decoder-based Indistinguishability along with strong CPA security and decoder-based index and message hiding, as defined in Section 6.2.*

8.1 Achieving Weak Decoder-based Indistinguishability from 0-bounded Indistinguishability

In this section, we show how we can compile a PSBEG with 0-bounded indistinguishability to a new PSBEG construction that achieves weak decoder-based indistinguishability, while maintaining all the other security properties.

$\underline{\mathsf{Setup}(1^\lambda, C)}$:

1. Fix any constants $\ell, \alpha, \beta, \gamma$ such that
$$0 < \beta < \alpha < 1 \quad , \quad 0 < \gamma < 1 \quad , \quad \ell \in \mathbb{N} \quad \text{and} \quad \ell(1-\gamma)(\alpha-\beta)^2 > 4 \ .$$
2. For $\zeta \in [\ell]$, sample $(\mathsf{pk}_\zeta, \mathsf{msk}_\zeta, \{\mathsf{sk}_{\zeta,i}\}_{i\in[n]}) \leftarrow\$ \mathsf{PSBEG.Setup}(1^\lambda, C)$.
3. Output $\mathsf{pk} := \{\mathsf{pk}_\zeta\}_{\zeta\in[\ell]}$, $\mathsf{msk} := \{\mathsf{msk}_\zeta\}_{\zeta\in[\ell]}$, and $\mathsf{sk}_i := \{\mathsf{sk}_{\zeta,i}\}_{\zeta\in[\ell]}$ for all $i \in [n]$.

$\underline{\mathsf{EncPK}(\mathsf{pk}, \mu)}$:

1. Sample $\zeta \leftarrow\$ [\ell]$, compute $\hat{\mathsf{ct}} \leftarrow\$ \mathsf{PSBEG.EncPK}(\mathsf{pk}_\zeta, \mu)$. Output $\mathsf{ct} := (\zeta, \hat{\mathsf{ct}})$.

$\underline{\mathsf{EncSK}(\mathsf{msk}, \mu, j, \mathcal{T})}$:

1. Sample $\zeta \leftarrow\$ [\ell]$, compute $\hat{\mathsf{ct}} \leftarrow\$ \mathsf{PSBEG.EncSK}(\mathsf{msk}_\zeta, \mu, j, \mathcal{T})$. Output $\mathsf{ct} := (\zeta, \hat{\mathsf{ct}})$.

$\underline{\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})}$:

1. Parse ct as $(\zeta, \hat{\mathsf{ct}})$, and sk_i as $\{\mathsf{sk}_{1,i}, \ldots, \mathsf{sk}_{\ell,i}\}$. Output $\mathsf{PSBEG.Dec}(\mathsf{sk}_{\zeta,i}, \hat{\mathsf{ct}})$.

$\underline{\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in S})}$:

1. Parse ct as $(\zeta, \hat{\mathsf{ct}})$. Output $\mathsf{PSBEG.Combine}(\mathsf{pk}_\zeta, \hat{\mathsf{ct}}, \{d_j\}_{j\in S})$.

Fig. 13. Our compiler for constructing a PSBE scheme PSBEG′ with weak Decoder-based Indistinguishability from PSBEG with only 0-bounded Indistinguishability. Note that our compiler preserves all the other properties, i.e. strong CPA security, decoder-based index hiding and message hiding.

Correctness. Correctness of the resulting scheme PSBEG′ follows trivially from correctness of the underlying scheme PSBEG.

CPA security. Theorem 29 below proves that the scheme PSBEG′ satisfies strong CPA security if PSBEG also satisfies strong CPA security.

Theorem 29. *For any* PPT *adversary* $\mathcal{A}$*, there exists another* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A},\mathsf{PSBEG}'}(\lambda) \leq \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B},\mathsf{PSBEG}}(\lambda)$$

Proof. The proof directly follows the proof of Theorem 16 in [DDP+25]. □

Weak Decoder-based Indistinguishability. We prove weak Decoder-based Indistinguishability of PSBEG' in the theorem below.

Theorem 30. *If* PSBEG *achieves* 0*-bounded indistinguishability, then* PSBEG' *achieves weak decoder-based Indistinguishability.*

Proof. The proof is essentially identical to the proof of [Zha25, Theorem 44], except for the following (trivial) changes:

- The term p_ζ for all $\zeta \in [\ell]$ is defined with respect to secret-key encryptions to $(0, \emptyset)$, i.e. $\mathsf{EncSK}(\mathsf{msk}, \mu_b, 0, \emptyset)$ (where $b \leftarrow\!\!\$\ \{0,1\}$ is sampled randomly).
- Since we are considering an encryption scheme instead of a key encapsulation mechanism, the decoder in the 0-bounded indistinguishability game is accompanied with a message μ for which it can distinguish between public and secret-key ciphertexts.

□

Decoder-based Index Hiding. As in [Zha25], this follows immediately from decoder-based index hiding of the underlying scheme PSBEG.

Decoder-based Message Hiding. As in [Zha25], this follows immediately from decoder-based message hiding of the underlying scheme PSBEG.

8.2 Constructing Risky and Limited Traitor Tracing for General Access Structures from Weak PSBE

We now present our compiler to construct Risky and Limited Traitor Tracing for General Access Structures from a weak PSBEG scheme. The compiler is essentially the same as that given in Section 6.3, but we can only achieve Risky and Limited tracing because of weak decoder-based Indistinguishability of the PSBEG scheme.

Correctness and strong CPA security of our compiled scheme follow directly from correctness and strong CPA of the PSBEG scheme: we refer the reader to Section 6.3 for proofs of the same.

Risky and Limited Traceability. We prove that the resulting scheme achieves risky and Limited traceability in the theorem below.

Theorem 31. *If* PSBEG *is a* (α, β, γ)*-weak PSBE (as in Definition 30), then the compiled scheme achieves* $(\alpha/2, \gamma)$*-risky Limited traceability.*

Proof. **False Trace Probability.** We can rely on chernoff bounds and adaptive q-bounded decoder-based set hiding of PSBEG to prove that the tracer never blames honest parties, exactly as in the proof of Theorem 12. Note that, this part of the proof does not rely on indistinguishability.

Correct Trace Probability. We essentially follow the proof of Theorem 12. We prove that if the decoder is sufficiently good, some user will be accused with reasonable probability, in each iteration

$\underline{\mathsf{KeyGen}(1^\lambda, C)}$:

1. Sample $(\mathsf{pk}, \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\!\!\$\ \mathsf{PSBEG.Setup}(1^\lambda, C)$, and output $(\mathsf{pk}, \mathsf{tk} := \mathsf{msk}, \{\mathsf{sk}_i\}_{i\in[n]})$.

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu)}$:

1. Output $\mathsf{PSBEG.EncPK}(\mathsf{pk}, \mu)$.

$\underline{\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})}$:

1. Output $\mathsf{PSBEG.Dec}(\mathsf{sk}_i, \mathsf{ct})$.

$\underline{\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in\mathcal{S}})}$:

1. Output $\mathsf{PSBEG.Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in\mathcal{S}})$.

$\underline{\mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, \mu_0, \mu_1)}$:

1. Let $W = \lambda \cdot (2n/\beta)^2$. Parse tk as msk.
2. Let $\mathcal{J} := \emptyset$ and $j = 1$. While $C(x_{\mathcal{J}}) = 0$ and $j < n$:
   (a) For $i = 0 \ldots n$:
      i. Let $\mathsf{count}_{i,j} := 0$.
      ii. For $\ell = 1$ to W:
         A. Sample $b_{i,j,\ell} \leftarrow \{0, 1\}$.
         B. If $i = 0$, let $\mathsf{ct}_{i,j,\ell} \leftarrow\!\!\$\ \mathsf{PSBEG.EncSK}(\mathsf{msk}, \mu_{b_{i,j,\ell}}, i, \emptyset)$.
         C. Else if $i > 0$, then define $\mathsf{ct}_{i,j,\ell} \leftarrow\!\!\$\ \mathsf{PSBEG.EncSK}(\mathsf{msk}, \mu_{b_{i,j,\ell}}, i, \mathcal{J})$.
         D. If $D(\mathsf{ct}_{i,j,\ell}) = b_{i,j,\ell}$, update $\mathsf{count}_{i,j} := \mathsf{count}_{i,j} + 1$.
      iii. Let $\hat{p}_{i,j} := \mathsf{count}_{i,j}/W$.
   (b) Let $\mathcal{J}'_j := \{i \mid \hat{p}_{i-1,j} - \hat{p}_{i,j} \geq \epsilon/4n\}_{i\in[n]}$. Set $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}'_j$ and $j \leftarrow j + 1$.
3. **Output** $\mathcal{J}$.

Fig. 14. Our compiler to construct a $(\epsilon := \alpha/2, \delta := \gamma)$-risky Limited Traitor tracing for general access structures from a (α, β, γ)-weak PSBE scheme PSBEG for the same access structure. Note the factor of two in ϵ is just an artifact of the slight difference in definitions between TTG and weak decoder indistinguishability as defined in [Zha25].

of the tracing algorithm. Moreover, by relying on adaptive q-bounded decoder-based set hiding of PSBEG, we can also show that the tracer never blames the same party in two different iterations, exactly as in the proof of Theorem 15. Combining the two statements, we get that the tracer will indeed find an authorized set of corrupt parties to blame.

For a decoder D and public key pk, recall that $\mathsf{Good}_\alpha(D, \mathsf{pk})$ means $2\times\Pr\left[D(\mathsf{ct}) = b : \substack{b\leftarrow\!\!\$\{0,1\} \\ \mathsf{ct}\leftarrow\!\!\$\mathsf{EncPK}(\mathsf{pk},\mu_b)}\right] - 1 \geq \alpha$. Similarly, $\mathsf{Good}'_\beta(D, \mathsf{msk})$ means

$$2 \times \Pr\left[D(\mathsf{ct}) = b : \substack{b\leftarrow\!\!\$\{0,1\} \\ \mathsf{ct}\leftarrow\!\!\$\mathsf{EncSK}(\mathsf{msk},\mu_b,0,\emptyset)}\right] - 1 \geq \beta.$$

If the event $\mathsf{Good}'_\beta(D, \mathsf{msk})$ occurs, we know that the box can decrypt messages encrypted to index - set pair $0, \emptyset$, i.e. $p_0^j = 1/2 + \beta/2$ for all $j \in [n]$. Then, by relying on adaptive q-query decoder-based message hiding, we can show that $p_{n,j}^D$ must be $\leq 1/2 + \mathsf{negl}(\lambda)$ for all $j \in [n]$, exactly as in the proof of Theorem 4.

Fonally, by (α, β, γ)-weak decoder-based indistinguishability, we know,

$$\Pr[\mathsf{GoodTr}] = \Pr[\mathsf{Good}'_\beta(D, \mathsf{msk})] \geq \gamma \Pr[\mathsf{Good}_\alpha(D, \mathsf{pk})] - \mathsf{negl}(\lambda) \ .$$

By observing that the probability of $\mathsf{GoodDec}$ is equal to that of $\mathsf{Good}_\alpha(D, \mathsf{pk})$ by definition, we get the theorem. □

9 Eliminating Riskiness and Limitedness in Tracing

9.1 Risk Mitigation Compiler

Figure 15 presents our compiler for eliminating risk. This compiler generalizes the risk mitigation compiler given in [Zha20, DDP+25] to general access structures. Note that the Trace algorithm is not given ϵ (the probability with which a decoder box 'works') as input since this is a Limited scheme, meaning that ϵ is implicit in the public parameters of the scheme.

$\underline{\mathsf{KeyGen}(1^\lambda, C)}$:

1. For $j \in [T]$, sample // $T = (\beta/\alpha) \cdot \omega(\log \lambda)$
$$(\mathsf{pk}_j, \mathsf{sk}_{j,1}, \ldots, \mathsf{sk}_{j,n}, \mathsf{tk}_j) \leftarrow\!\!\$\ \mathsf{TTG}_{\mathsf{risky,Limited}}.\mathsf{KeyGen}(1^\lambda, C)$$
2. Output $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_T)$, $\mathsf{tk} = (\mathsf{tk}_1, \ldots, \mathsf{tk}_T)$ and $\mathsf{sk}_i = (\mathsf{sk}_{1,i}, \ldots, \mathsf{sk}_{T,i})$ for all $i \in [n]$.

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu)}$:

1. Sample $j \leftarrow\!\!\$\ [T]$. Output $(j, \mathsf{TTG}_{\mathsf{risky,Limited}}.\mathsf{Enc}(\mathsf{pk}_j, \mu))$.

$\underline{\mathsf{Dec}(\mathsf{sk}_i, (j, c))}$:

1. Output $d_i = \mathsf{TTG}_{\mathsf{risky,Limited}}.\mathsf{Dec}(\mathsf{sk}_{j,i}, c)$.

$\underline{\mathsf{Combine}(\mathsf{pk}, (j, c), \{d_i\}_{i \in \mathcal{S}})}$:

1. Output $\mathsf{TTG}_{\mathsf{risky,Limited}}.\mathsf{Combine}(\mathsf{pk}_j, c, \{d_i\}_{i \in \mathcal{S}})$.

$\underline{\mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, \mu_0, \mu_1)}$:

1. For $j \in [T]$, let D_j be the decoder $D_j(c, \mu_0, \mu_1) := D((j, c), \mu_0, \mu_1)$.
2. For $j \in [T]$, run $\mathcal{J}_j^* \leftarrow\!\!\$\ \mathsf{TTG}_{\mathsf{risky,Limited}}.\mathsf{Trace}^{D_j(\cdot)}(\mathsf{pk}_j, \mathsf{tk}_j, \mu_0, \mu_1)$.
3. Output $\mathcal{J}^* = \bigcup_{j \in [T]} \mathcal{J}_j^*$.

Fig. 15. A generic compiler to get a β-risky Limited traitor tracing scheme $\mathsf{TTG}_{\mathsf{Limited}}$ for general access structures from an α-risky Limited traitor tracing scheme $\mathsf{TTG}_{\mathsf{risky,Limited}}$ for the same access structure. Setting $\beta = 1$ gives us a Limited traitor tracing scheme for general access structures.

Correctness. Correctness of this compiler follows trivially from correctness of the underlying scheme $\mathsf{TTG}_{\mathsf{risky,Limited}}$. Specifically, for any $j \in [T]$, an honestly generated ciphertext with respect to the jth keys will be correctly decrypted by any authorized set of parties. Moreover, observe that the size of public and secret keys only grows by a factor of T, which is independent of the number of parties.

Semantic security. Theorem 32 below proves semantic security of the compiled scheme $\mathsf{TTG}_{\mathsf{Limited}}$ by relying on semantic security of the underlying scheme $\mathsf{TTG}_{\mathsf{risky,Limited}}$.

Theorem 32. *For any* PPT *adversary* $\mathcal{A}$*, there exists another* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A},\mathsf{TTG}_{\mathsf{Limited}}}(\lambda) \leq \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B},\mathsf{TTG}_{\mathsf{risky,Limited}}}(\lambda).$$

Proof. The proof is a direct generalization of the proof of Theorem 19 in [DDP+25]. □

Traceability. Theorem 33 below proves that the compiler results in a β-risky Limited scheme. This implies that by setting $\beta = 1$, we can indeed eliminate risk via this compiler.

Theorem 33. *Assume* $T = (\beta/\alpha) \cdot \omega(\log \lambda)$*. If* $\mathsf{TTG}_{\mathsf{risky,Limited}}$ *is an* α*-risky Limited traitor tracing scheme for general access structures, then* $\mathsf{TTG}_{\mathsf{Limited}}$ *is a* β*-risky Limited traitor tracing scheme for the same access structure.*

Proof (Sketch). The proof is the same as the proof for Theorem 10 in [Zha20]. Roughly speaking, since the adversary in the tracing game is allowed to access the secret keys of all n parties, the access structure C does not matter. Hence, we only give a proof sketch below, and refer the reader to [Zha20] for the full proof.

Consider a decoder D^* outputted by an adversary. We say that $j \in [T]$ is "good" if D^*_j has a high chance of decrypting ciphertexts for index j of $\mathsf{TTG}_{\mathsf{risky,Limited}}$. D^* can only decrypt ciphertexts for j where D^*_j is good; thus the event $\mathsf{GoodDec}$ (for the compiled scheme $\mathsf{TTG}_{\mathsf{Limited}}$) implies that the fraction of good j is large. Since each j represents a different instance of the risky Limited scheme, each of the decoders D^*_j should intuitively have an α chance of being traced to an authorized subset of users. As long as the number of good j is larger than $\omega(\log \lambda)/\alpha$, then we would expect that, with overwhelming probability, at least one of the D^*_j traces. One challenge is that, for general decoders with arbitrary inverse-polynomial success probability, the number of good j may be small. In particular, if D^* has advantage $1/T$, then it could be that only a single j is good. We resolve this by only guaranteeing *Limited* security, which implies that a constant fraction of j are good. Another challenge is that the attacker can choose adaptively which of the j will good and hence traceable, so the tracing probabilities are not independent events. Nevertheless, by a careful analysis as in [Zha20], we can prove security. □

9.2 Limitedness Mitigation Compiler

Figures 16 and 17 present our compiler for eliminating Limitedness. This compiler essentially generalizes the compiler given in [DDP+25] from threshold to general access structures.

We start with describing the Goldreich-Levin Decoding algorithm which forms a crucial part of the compiler.

Theorem 34 (Goldreich-Levin Decoding [GL89]). *There exists a constant* Γ *and oracle algorithm* $\mathsf{GL}^D(\ell, \epsilon')$ *running in time* $\mathsf{poly}(\ell, \log(1/\epsilon'))$ *and making* $\mathsf{poly}(\ell, \log(1/\epsilon'))$ *queries to* D*, such that the following holds. If there exists an* $x \in \{0,1\}^\ell$ *such that* $\Pr[D(r) = x \cdot r \bmod 2 : r \leftarrow\$ \{0,1\}^\ell] \geq 1/2 + \epsilon'$*, then* $\Pr[\mathsf{GL}^D(\ell, \epsilon') = x] \geq \Gamma \times (\epsilon')^2$.

Correctness. Correctness follows almost directly from correctness of the underlying scheme $\mathsf{TTG}_{\mathsf{Limited}}$. In more detail, for all $u \in [y], v \in [z]$, $\mu'_{u,v} = \mu_{u,v}$ with overwhelming probability, by correctness of $\mathsf{TTG}_{\mathsf{Limited}}$. This means that, for all $v \in [z]$, $\mu'_v = \mu_v$, and, $b'_v = b_v$, which implies $k' = k$. Hence, $\mu' = (\mu \oplus k) \oplus k' = \mu$.

$\underline{\mathsf{KeyGen}(1^\lambda, C)}$:

1. Run $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{tk}) \leftarrow\$ \mathsf{TTG}_{\mathsf{Limited}}.\mathsf{KeyGen}(1^\lambda, C)$. Output $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{tk})$.

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu)}$:

1. Let $y = \omega(\log \lambda)$ and $z = z(\lambda)$ be a polynomial. // $\mu \in \{0,1\}^z$
2. For all $u \in [y], v \in [z]$, sample $\mu_{u,v} \leftarrow\$ \mathcal{M}$ and compute $c_{u,v} \leftarrow\$ \mathsf{TTG}_{\mathsf{Limited}}.\mathsf{Enc}(\mathsf{pk}, \mu_{u,v})$.
3. For all $v \in [z]$, compute $\mu_v := \mu_{1,v} \oplus \mu_{2,v} \ldots \oplus \mu_{y,v}$.
4. Sample $s \leftarrow\$ \mathcal{M}$, and compute $b_v := \langle s, \mu_v \rangle \bmod 2$ for all $v \in [z]$. // $\langle s, \mu \rangle$ denotes bit-wise inner-product
5. Define $k := (b_1, \ldots, b_z) \in \{0,1\}^z$. Output $c = (\mu \oplus k, s, \{c_{u,v}\}_{u \in [y], v \in [z]})$.

$\underline{\mathsf{Dec}(\mathsf{sk}_i, c)}$:

1. Parse c as $(\hat{c}, s, \{c_{u,v}\}_{u \in [y], v \in [z]})$.
2. For all $u \in [y], v \in [z]$, compute $d_{u,v,i} := \mathsf{TTG}_{\mathsf{Limited}}.\mathsf{Dec}(\mathsf{sk}_i, c_{u,v})$.
3. Output $d_i := \{d_{u,v,i}\}_{u \in [y], v \in [z]}$.

$\underline{\mathsf{Combine}(\mathsf{pk}, c, \{d_j\}_{j \in \mathcal{S}})}$:

1. Parse c as $(\hat{c}, s, \{c_{u,v}\}_{u \in [y], v \in [z]})$ and d_i as $\{d_{u,v,i}\}_{u \in [y], v \in [z]}$ for all $i \in \mathcal{S}$.
2. For all $u \in [y], v \in [z]$, compute $\mu'_{u,v} \leftarrow \mathsf{TTG}_{\mathsf{Limited}}.\mathsf{Combine}(\mathsf{pk}, c_{u,v}, \{d_{u,v,i}\}_{i \in \mathcal{S}})$.
3. For all $v \in [z]$, compute $\mu'_v := \mu'_{1,v} \oplus \ldots \oplus \mu'_{y,v}$, and $b'_v := \langle s, \mu'_v \rangle \bmod 2$.
4. Define $k' := (b'_1, \ldots, b'_z) \in \{0,1\}^z$, and output $\mu' := \hat{c} \oplus k'$.

Fig. 16. A generic compiler to get a traitor tracing scheme for general access structures TTG from a limited traitor tracing scheme $\mathsf{TTG}_{\mathsf{Limited}}$ for the same access structure. Here the message space for $\mathsf{TTG}_{\mathsf{Limited}}$ is assumed to be $\mathcal{M} := \{0,1\}^\ell$. The Trace algorithm is described in Fig. 17.

$\underline{\mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^{1/\epsilon}, \mu_0, \mu_1)}$:

1. Construct algorithm $D_1(s, \{c_u\}_{u \in [y]})$ as follows:
   - Sample random $v \leftarrow\$ [z]$. For all $u \in [y]$, set $c_{u,v} := c_u$.
   - For all $v' \in [z] \setminus \{v\}$ and all $u \in [y]$, sample $\mu_{u,v'} \leftarrow\$ \mathcal{M}$ and compute $c_{u,v'} \leftarrow\$ \mathsf{TTG}_{\mathsf{Limited}}.\mathsf{Enc}(\mathsf{pk}, \mu_{u,v'})$.
   - For all $v' \in [z] \setminus \{v\}$, compute $\mu_{v'} := \mu_{1,v'} \oplus \ldots \oplus \mu_{y,v'}$.
   - For all $v' > v$, set $b_{v'} := \langle s, \mu_{v'} \rangle \bmod 2$. For all $v' \leq v$, sample $b_{v'} \leftarrow\$ \{0,1\}$. Define $k := (b_1, \ldots, b_z)$.
   - Sample $\beta \leftarrow\$ \{0,1\}$, and output $\beta \oplus b_v \oplus D(\mu_\beta \oplus k, s, \{c_{u,v}\})$.
2. Construct $D_2(\ \{c_u\}_{u \in [y]}\ ) := \mathsf{GL}^{D_1(\cdot, (c_u)_{u \in [y]})}(\ell, \epsilon' = \epsilon/z)$.
3. Sample $\hat{\mu}_0, \hat{\mu}_1 \leftarrow\$ \mathcal{M}$ and construct $D_3(c)$ as follows:
   - For $w \in 1, \ldots, \zeta := (2yz^3/\Gamma\epsilon^3) \times \omega(\log \lambda)$:
     - Choose a random $u \in [y]$, and set $c_u = c$.
     - Then for each $u' \in [y] \setminus \{u\}$, sample $\mu_{u'} \leftarrow\$ \mathcal{M}$ and run $c_{u'} \leftarrow\$ \mathsf{TTG}_{\mathsf{Limited}}.\mathsf{Enc}(\mathsf{pk}, \mu_{u'})$.
     - Run $\mu' \leftarrow D_2(\ \{c_u\}_{u \in [y]}\ )$, and set $\mu^{(z)} := \mu' \oplus \mu_1 \ldots \oplus \mu_{u-1} \oplus \mu_{u+1} \ldots \oplus \mu_y$.
   - If $\mu^{(w)} = \hat{\mu}_0$ for any $w \in [\zeta]$, output 0. Otherwise, output 1.
4. Return $\mathsf{TTG}_{\mathsf{Limited}}.\mathsf{Trace}^{D_3(\cdot)}(\mathsf{tk}, \mathsf{pk}, \hat{\mu}_0, \hat{\mu}_1)$.

Fig. 17. The Trace algorithm for our generic compiler to get a traitor tracing scheme for general access structures TTG from a limited traitor tracing scheme $\mathsf{TTG}_{\mathsf{Limited}}$ for the same access structure.

Also note that the public key and secret keys of the resulting scheme are of the same size as the original scheme. Moreover, the ciphertext size grows by a factor of $y \cdot z$ which is $\mathsf{poly}(\lambda)$.

Ssemantic Security. Theorem 35 below proves semantic security of the compiled scheme by relying on semantic security of the underlying Limited scheme.

Theorem 35. *For any* PPT *adversary* $\mathcal{A}$*, there exists another* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A},\mathsf{TTG}}(\lambda) \leq 4 \cdot z \cdot \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B},\mathsf{TTG}_{\mathsf{Limited}}}(\lambda),$$

where $z = z(\lambda)$ *is a parameter of the scheme.*

Proof. The proof is the same as that for Theorem 22 in [DDP+25]. □

Traceability. Theorem 36 below proves that the resulting scheme indeed achieves traceability.

Theorem 36. *For* $y = \omega(\log \lambda), \epsilon' = \epsilon/z, \zeta = (2yz^3/\Gamma\epsilon^3) \times \omega(\log \lambda)$*. Suppose* $\ell = \omega(\lambda)$*. If* $\mathsf{TTG}_{\mathsf{Limited}}$ *is a secure Limited traitor tracing scheme for general access structures, then* TTG *is a secure (not-Limited) traitor tracing scheme for the same access structure.*

Proof. The proof is the same as that for Theorem 23 in [DDP+25]. Crucially, since the adversary can potentially corrupt upto n parties, the access structure plays no role in the proof. □

10 Candidate TTT with Small Ciphertexts from Obfuscation

Here, we give a candidate TTT which traces a nearly optimal number of users with short ciphertexts. We prove security in an idealized model for obfuscation, and leave proving the security of our construction under standard obfuscation definitions such as indistinguishability obfuscation (iO) as an intriguing open question. The ingredients for our construction will be:

- A digital signature scheme $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$
- An obfuscation scheme Obf. On a deterministic program P, we will heuristically model $\mathsf{Obf}(P)$ as an oracle that implements P.

For simplicity, we first describe a scheme with large public keys and ciphertexts which allows us to describe our new tracing algorithm. Then we will explain how to shrink the parameters.

Guided by the lower-bound in Theorem 8, we know we cannot trace all users. Instead, we will chose a polynomial λ^c, and will only guarantee the ability to trace $a = (1 - \lambda^{-c})t$ users. The ciphertext size will grow polynomially with λ^c, so c must be an a priori fixed constant. This matches the lower bound of Theorem 8, which shows that ciphertexts for such a scheme must have size at least $\Omega(a/(t-a)) = \Omega(\lambda^c)$. This constant c will not be apparent yet, but we will make use of it later, when we explain how to make our scheme have small ciphertexts.

Finally, we will always assume that $t \geq \lambda^{c'}$ for some $c' > c$. If $t \leq \lambda^c$, then we can always trace using a threshold PSBE scheme with adaptive security (by relying on the generic compilers given in Section 6.3), which can be straightforwardly obtained from ideal obfuscation with ciphertext size $\mathsf{poly}(t, \lambda) = \mathsf{poly}(\lambda)$.

We first describe how to generate keys, encrypt, and decrypt:

- $\mathsf{KeyGen}(1^\lambda, 1^n, 1^t, 1^a)$: For $j = 1, \cdots, n$, run $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \mathsf{Gen}(1^\lambda)$. Let $\mathsf{pk} = (\mathsf{pk}_1, \cdots, \mathsf{pk}_n)$ and $\mathsf{tk} = \{\}$.
- $\mathsf{Enc}(\mathsf{pk}, \mu)$: Let $O : [n] \to \{0,1\}$ be the all-1's function $O(j) = 1$. Choose a random string $\tau \in \{0,1\}^\lambda$. Let $P_{\tau,O}(\mathcal{J}, \{d_j\}_{j\in\mathcal{J}})$ be the following program:
  - If $\mathsf{Ver}(\mathsf{pk}_j, \tau, d_j)$ rejects for any j, abort and output $\perp$.
  - If for each $j \in \mathcal{J}$, check if $O(j) = 1$. If any $O(j) = 0$, abort and output $\perp$.
  - Otherwise, output μ.

  Let $\hat{P} = \mathsf{Obf}(P_{\tau,O})$. Output $\mathsf{ct} = (\tau, \hat{P})$.
- $\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct})$: output $d_j \leftarrow \mathsf{Sig}(\mathsf{sk}_i, \tau)$
- $\mathsf{Combine}(\mathsf{pk}, \mathsf{ct}, \{d_j\}_{j\in\mathcal{J}})$: run and output $\mu \leftarrow \hat{P}(\mathcal{J}, \{d_j\}_{j\in\mathcal{J}})$.

$\mathsf{Trace}^D(\mathsf{pk}, \mathsf{tk}, 1^{1/\epsilon}, \mu_0, \mu_1)$. We now develop some tools that will enable us to trace. For a subset $S \subseteq [n]$, index $j^* \in [0, n] \setminus S$, and integer $\ell \in [|S|, n]$, let $\mathcal{D}_{S,j^*,\ell}$ be the distribution over oracles $O : [n] \to \{0, 1\}$ where:

- $O(j) = 1$ for all $j \in S$
- $O(j^*) = 0$ if $j^* > 0$
- $O(j)$ for $j \notin S \cup \{j^*\}$ are uniform, conditioned on $\sum_{j \in [n]} O(j) = \ell$.

By setting $j^* = 0$, we just condition on $O(j) = 1$ for $j \in S$ and $\sum_{j \in [n]} O(j) = \ell$, and ignore the condition on $O(j^*)$.

Let $p_{S,j^*,\ell}$ be the probability that $D(\mathsf{ct} = (\tau, \hat{P})) = b$, where b is uniform in $\{0, 1\}$ and $\hat{P}$ is the obfuscation of $P_{\tau,O}$ as in Enc, except that we sample $O \leftarrow \mathcal{D}_{S,j^*,\ell}$. We can approximate $p_{S,j^*,\ell}$ to arbitrary inverse-polynomial error by computing many samples and taking the empirical probability. For simplicity, we will assume we can compute $p_{S,j^*,\ell}$ exactly. We now describe our tracing algorithm:

- Initialize sets $S, A = \{\}$ and integer $\ell = n$.
- Repeat the following until $|A| \geq a$: if there is any $j \in [n] \setminus S$ such that $p_{S \cup \{j\},0,\ell} \geq p_{S,j,\ell-1} + \epsilon/2n$:
  - Add all such j to A. Then find a $j' \notin S$ such that $p_{S \cup j',0,\ell} \geq p_{S,0,\ell}$, and update $S = S \cup \{j'\}$ and $\ell = \ell$.
  - Otherwise if there is no such j, let $\ell = \ell - 1$.
- Output A.

Intuition. The check for $p_{S \cup \{j\},0,\ell} \geq p_{S,j,\ell-1} + \epsilon/2n$ is basically checking if turning any user j on ($O(j) = 1$) vs off ($O(j) = 0$) independently of the other users causes a significant change in success probability. For any honest user, there will be a minimal change in success probability, so we can accuse any user with a noticeable change. We will show that, if we don't have enough users, we can add a user to S (in the case where we accuse a new user) or decrease ℓ (in the case where we don't accuse any new users) while still maintaining a high success probability of the decoder. Eventually, this process is guaranteed to accuse a users. The key to ultimately obtaining a scheme with short ciphertexts independent of t is to ensure that we accuse a users while maintaining a very large ℓ.

Analysis. Throughout, we will model the obfuscated program $\hat{P}$ as an oracle for P. Let Q be the number of queries that the decoder box makes to the oracle when given a ciphertext as input. We now invoke the security of the signature scheme:

Lemma 18. *Suppose the underlying signature scheme* $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *is secure. Suppose we model* $\hat{P}$ *as an oracle for* P*. Suppose user* j *is honest, meaning the adversary never queried for a secret key on* j*. Then, except with negligible probability,* $D(\mathsf{ct} = (\tau, \hat{P}))$ *never makes a query to* $\hat{P}$ *on a subset* $\mathcal{J}$ *such that* $j \in \mathcal{J}$*.*

Proof. Suppose D made such a query. Then in particular D must have provided a signature on τ relative to pk_j. But since the adversary never obtained a secret key for pk_j, a straightforward reduction to the security of the underlying signature scheme shows that such query would break security. □

Going forward, we will therefore always assume that the adversary only queries the oracle for P on $\mathcal{J}$ which are all dishonest. We call this the "good query" condition.

We first prove that honest users are never accused.

Lemma 19. *Under the "good query" condition, for any set S and any honest user $j^* \notin S$, $p_{S\cup\{j^*\},0,\ell} = p_{S,j^*,\ell-1}$. In particular, j^* will never be accused.*

Proof. Let O' be the restriction of O to $[n] \setminus \{j^*\}$. Then $\mathcal{D}_{S\cup\{j^*\},0,\ell}$ and $\mathcal{D}_{S,j^*,\ell-1}$ yield identical distributions on the restriction O': both set $O(j) = 1$ for $j \in S$, and then uniformly sample $\ell-1-|S|$ elements outside of $S\cup\{j^*\}$ to be 1. Under the good query condition, queries to P can be answered using O' instead of O. Hence, each query is answered identically under $\mathcal{D}_{S\cup\{j^*\},0,\ell}$ and $\mathcal{D}_{S,j^*,\ell-1}$, meaning that the decoder box cannot distinguish between these two distributions, implying that $p_{S\cup\{j^*\},0,\ell}$ cannot be far from $p_{S,j^*,\ell-1}$. This proves the theorem. □

Now we need to show that Trace terminates in polynomial time and succeeds in accusing at least a users. We first show that for small ℓ, $p_{S,0,\ell}$ is tiny.

Lemma 20. *Under the "good query" condition, $p_{S,0,\ell} \le 1/2 + Q \times ((\ell - |S|)/(n - |S|))^{t-|S|}$*

Proof. We call a query to the oracle for a set $\mathcal{J}$ "bad" if $O(j) = 1$ for all $j \in \mathcal{J}$. Under the "good query" condition, if there are no bad queries, then the decoder learns nothing about the encrypted message. Therefore, $p_{S,0,\ell}$ can be upper-bounded by $1/2 + g$, where g is the probability that there is some bad query. Consider some fixed sequence of queries on $\mathcal{J}_1, \cdots, \mathcal{J}_Q$. For each $\mathcal{J}_k$, there are $\binom{n-|S|}{\ell-|S|}$ possible choices of oracle O from $\mathcal{D}_{S,0,\ell}$, and $\binom{n-|\mathcal{J}_k\cup S|}{\ell-|\mathcal{J}_k\cup S|}$ ways to choose O such that $O(j) = 1$ for all $j \in S$ and all $j \in \mathcal{J}_k$. Therefore, the probability that the set $\mathcal{J}_k$ is "bad" is at most

$$\begin{aligned}\binom{n - |\mathcal{J}_k \cup S|}{\ell - |\mathcal{J}_k \cup S|} \Big/ \binom{n - |S|}{\ell - |S|} &= \frac{(\ell - |S|)(\ell - |S| - 1)\cdots(\ell - |\mathcal{J}_k \cup S| + 1)}{(n - |S|)(n - |S| - 1)\cdots(n - |\mathcal{J}_k \cup S| + 1)} \\ &\le \left(\frac{\ell - |S|}{n - |S|}\right)^{|\mathcal{J}_k\cup S|-|S|} \le \left(\frac{\ell - |S|}{n - |S|}\right)^{t-|S|}\end{aligned}$$

The last inequality uses the fact that $\mathcal{J}_k = t$ hence $|\mathcal{J}_k \cup S| \ge t$, and $\ell \le n$. Summing over all Q possible queries gives the lemma. □

On the other hand, we will show that for all S, ℓ considered by the tracing algorithm, $p_{S,0,\ell}$ stays not too small.

Lemma 21. *Assume the "good query" condition. Then* Trace *maintains the following invariants: (1) it is always able to find a j' in the case it adds a user to A, (2) $|S| \le |A|$, and (3) $p_{S,0,\ell} \ge 1/2 + \epsilon(1 - (n - \ell)/2n) \ge 1/2 + \epsilon/2$*

Proof. We prove by induction. In the base case, $S = A = \{\}$ and $\ell = n$, we have that $p_{\{\},0,n}$ corresponds to the success probability of the decoder on honest ciphertexts, and so $p_{\{\},0,n} \ge 1/2 + \epsilon = 1/2 + \epsilon(1 - (n - \ell)/2n)$. For the inductive step, suppose at the beginning of an iteration we have S, ℓ such that $p_{S,0,\ell} \ge 1/2 + \epsilon(1 - (n - \ell)/2n)$. In particular, $p_{S,0,\ell} \ge 1/2 + \epsilon/2$.

Since we didn't terminate yet, at the beginning of the iteration we must have $|S| \le |A| < a \le t$. This allows us to conclude that $\ell > |S|$: otherwise, by Lemma 20, $p_{S,0,\ell} \le 1/2 + Q \times 0^{t-|S|} = 1/2$, contradicting our inductive assumption. To prove (1), we observe that as long as $\ell > |S|$, we can sample from $\mathcal{D}_{S,0,\ell}$ by choosing a random $j \notin S$, and sampling $\mathcal{D}_{S\cup\{j\},0,\ell}$. Therefore, $p_{S,0,\ell} = \mathbb{E}_{j\notin S}[p_{S\cup\{j\},0,\ell}]$. Thus, there is some $j' \notin S$ such that $p_{S\cup\{j'\},0,\ell} \ge p_{S,0,\ell}$. In the case where a (dishonest) user is accused, such a j' will therefore be found.

To prove (2), notice that if we add a user to S, we add at least one user to A, and therefore since S started out no larger than A, this remains the case.

To prove (3), we consider two cases: in the first case, in this iteration, some (dishonest) user is accused. Here, we update S to $S' = S \cup \{j'\}$ and $\ell' = \ell$, where j' is the user gauranteed above for this case. Thus, $p_{S',0,\ell'} = p_{S\cup\{j\},0,\ell} \geq p_{S,0,\ell} = 1/2 + \epsilon(1 - (n-\ell)/2n) = 1/2 + \epsilon(1 - (n-\ell')/2n)$, as desired.

In the second case, no user is accused. Here, notice that we can sample from $\mathcal{D}_{S,0,\ell-1}$, which must have at least one 0 outside S, by choosing a random $j \notin S$, and then sampling from $\mathcal{D}_{S,j,\ell-1}$. Thus,

$$p_{S,0,\ell-1} = \mathbb{E}_{j\notin S}[p_{S,j,\ell-1}] \geq \mathbb{E}_{j\notin S}[p_{S\cup\{j\},0,\ell}] - \epsilon/2n = p_{S,0,\ell} - \epsilon/2n$$

The second inequality follows from the fact that no party was accused in this iteration, i.e. for all $j \notin S$, $p_{S\cup\{j\},0,\ell} < p_{S,j,\ell-1} + \epsilon/2n$.

In this case, we update S to $S' = S$ and ℓ to $\ell' = \ell - 1$. Thus we have that $p_{S',0,\ell'} = p_{S,0,\ell-1} \geq p_{S,0,\ell} - \epsilon/2n \geq 1/2 + \epsilon(1 - (n-\ell+1)/2n) = 1/2 + \epsilon(1 - (n-\ell')/2n)$, as desired. □

We are now ready to prove that the tracing algorithm terminates and accuses at least a users. In each iteration, either S increases by 1 or ℓ decreases by 1; thus $\ell - |S|$ always decreases by 1. Notice $\ell - |S|$ is initially n. As $\ell - |S|$ can never be negative, this means the number of iterations is at most n. Therefore the algorithm terminates and does so in an efficient number of steps. Since the algorithm only terminates once at least a users are accused, this means Trace eventually accuses at least a users.

The above argument actually readily extends to setting $a = t$, allowing t users to be accused. However, when it comes time to compress the ciphertexts below, it will be crucial that ℓ remains rather large. To accuse the full t users, in fact ℓ will become rather small, as a consequence of our lower-bound in Theorem 8. However, we now explain that if a is slightly lower than t, namely $a = t(1 - \lambda^{-c})$, then ℓ remains rather large. Indeed, by Lemma 20, the final ℓ, S satisfy

$$\begin{aligned} p_{S,0,\ell} &\leq 1/2 + Q \times ((\ell - |S|)/(n - |S|))^{t-|S|} \\ &< 1/2 + Q \times ((\ell - |S|)/(n - |S|))^{t\lambda^{-c}} \\ &\leq 1/2 + Q \times ((\ell - |S|)/(n - |S|))^{\lambda^{c'-c}} \end{aligned}$$

where we recall that c' is the constant $c' > c$ such that we assumed $t \geq \lambda^{c'}$.

On the other hand, Lemma 21 shows that $p_{S,0,\ell} \geq 1/2+\epsilon/2$. This means that $\ell/n \geq (\ell-|S|)/(n-|S|) \geq (\epsilon/2Q)^{\lambda^{-(c'-c)}} \geq 1 - \lambda^{-(c'-c)} \ln(2Q/\epsilon)$. Since Q is polynomial and ϵ is inverse polynomial, we can absorb the log factors into an arbitrarily small polynomial term, obtaining $\ell/n \geq 1 - O(\lambda^{-c''})$ for some constant $c'' > 0$.

Note that, in contrast if we allowed $a = t$, then an adaptation of Theorem 8 would show that ℓ may need to be as small as $\approx t$.

Compressing the ciphertexts. We now explain how to achieve short ciphertexts. There are a few reasons the scheme above naively has large ciphertexts:

- The program P needs to have the signature verification keys pk_j for each user hard-coded. This gives a lower-bound of $\Omega(n)$ on the ciphertext length.

– The program P has the function O hard-coded. The truth-table for O requires n bits.
– The input length is $\Omega(n)$ bits, so if using circuit obfuscation, the circuits must have $\Omega(n)$ gates, giving a lower-bound of $\Omega(n)$ on the ciphertext length.

We now explain how to eliminate each of these constraints:

– We first compress the n verification keys using a simple signature delegation mechanism. To do so, we will have a single master signature public key/secret key pair $\mathsf{mpk}/\mathsf{msk}$, and set the individual public keys to be $\mathsf{pk}_j = (\mathsf{mpk}, j)$. The individual secret key will be a tuple $(\mathsf{pk}'_j, \mathsf{sk}'_j, \sigma_j)$ where $(\mathsf{pk}'_j, \mathsf{sk}'_j) \leftarrow \mathsf{Gen}(1^\lambda)$ is an independent key pair, and $\sigma_j \leftarrow \mathsf{Sig}(\mathsf{msk}, \mathsf{pk}'_j)$. A signature on τ relative to (mpk, j) will be a tuple $d_j = (\mathsf{pk}'_j, \sigma_j, \sigma'_j)$ where $\sigma'_j \leftarrow \mathsf{Sig}(\mathsf{sk}'_j, \tau)$. Now the program P only needs to store mpk, which immediately lets it verify all of the signatures.
– In order to make our tracing algorithm work, we need to sample from the distribution $\mathcal{D}_{S,j^*,\ell}$ for various S, j^*, ℓ. Instead of sampling uniform O subject to the constraints that $O(j) = 1$ for $j \in S$, $O(j^*) = 0$, and $\sum_j O(j) = 1$, we will instead *pseudorandomly* sample such O. That is, we will assume we have a PRF $\mathsf{F}(k, x)$ which outputs a suitably discretized random number in $[0, 1]$. To pseudorandomly sample from $\mathcal{D}_{S,j^*,\ell}$, we will choose PRF keys k_i for $i = 1, \cdots, \lambda^c$, and set $O_i(j) = \begin{cases} 1 & \text{if } \mathsf{F}(k_i, j) < \alpha \\ 0 & \text{otherwise} \end{cases}$. Here, $\alpha = ((\ell - |S|)/(n - |S|))^{1/\lambda^c}$. If $O_i(j) = 0$ for any $j \in S$ or $O_i(j^*) = 1$, we will discard that k_i and try again. The expected number of trials for each k_i is $\alpha^{-|S|}$, so this process is efficient as long as $\alpha^{|S|}$ is inverse polynomial.
After sampling $k_1, \cdots, k_{\lambda^c}$, we define $O(j) = \wedge_i O_i(j)$. By our construction of O_i, $O(j) = 1$ for $j \in S$ and $O(j^*) = 0$. Moreover, for $j \notin S \cup \{j^*\}$, $O_i(j) = 1$ with probability α, and so $O(j) = 1$ with probability $\alpha^{\lambda^c} = ((\ell - |S|)/(n - |S|))$. Therefore, $\mathbb{E}[\sum_j O(j)] = |S| + ((\ell - |S|)/(n - |S|)) \times (n - |S|) = \ell$. The expectation moreover is the mode, and therefore occurs with probability at least $1/n$. Therefore, we check if $\sum_j O(j) = \ell$, and if not, discard $k_1, \cdots, k_{\lambda^c}$, and try again from the beginning. The expected number of trials is n, and so this process is efficient.
If we model $\hat{P}$ as an oracle for P, then using the pseudorandomness of F, for each sample of k_i, we can replace $\mathsf{F}(k_i, \cdot)$ with a truly random function. The result is that each O_i comprises truly iid samples from the Bernoulli distribution, where each sample is 1 with probability α. Once we finally construct O by conditioning on the various events, it is therefore distributed exactly as $\mathcal{D}_{S,j^*,\ell}$.
It remains to show that we can take $\alpha^{|S|} = ((\ell - |S|)/(n - |S|))^{|S|/\lambda^c}$ to be inverse polynomial to maintain efficiency. This is where the large-ℓ lower-bound comes in. Recall that we can take $((\ell - |S|)/(n - |S|)) \geq (\epsilon/2Q)^{\lambda^c/t}$. Therefore, $\alpha^{|S|} = ((\ell - |S|)/(n - |S|))^{|S|/\lambda^c} \geq (\epsilon/2Q)^{|S|/t} \geq (\epsilon/2Q)^{a/t} \geq \epsilon/2Q$, which is inverse polynomial as desired.
– Instead of obfuscating circuits, we will use Turing machine obfuscation. Since at this point the description of P is small, it can be implemented by a small Turing machine. Note that ideal circuit obfuscation implies ideal Turing machine obfuscation using well-known ideas [BCP14, ABG+13]. At a very high-level, each intermediate state of the computation is encrypted and authenticated. Circuit obfuscation is then applied to the program that decrypts and checks the authentication, performs a single step of the computation, and then re-encrypts and re-authenticates; the final output is given in the clear rather than encrypted. Evaluating the Turing machine is accomplished by repeatedly evaluating the obfuscated circuit program until the final output is produced. Then, modeling the underlying circuit obfuscation as an oracle, it is straightforward to show that the view of any adversary can be simulated just given black-box access to the Turing machine.

10.1 Challenges for Provable Security

Here, we discuss what seem to be challenges in turning our candidate into a provably-secure construction. We first start with a seemingly minor obstacle, which we believe is not the main bottleneck.

The Turing Machine Bottleneck. First, we used obfuscation for Turing machines, but direct obfuscation constructions are for circuits. In the ideal/virtual black box setting where the obfuscated program is heuristically treated as an oracle, Turing machine obfuscation follows readily from circuit obfuscation. However, for the standard notion of indistinguishability obfuscation (iO), existing constructions of Turing machine obfuscation (e.g. [AJS17]) restrict the input to a bounded size L, and the obfuscated program grows polynomially in L. In our case, the input to P is a list of t decryption values, and so existing solutions would result in a program growing with t.

We believe this limitation is not fundamental, though we omitted attempts to resolve it due to the more fundamental difficulties below making it unclear what a provable construction would look like. First, one can move the obfuscated program from ciphertext into the public key; now the ciphertext encrypts the hard-coded $k_1, \cdots, k_{\lambda^c}$ as well as the message μ using a CCA secure encryption scheme. The obfuscated program in the public key contains the decryption key, decrypts the ciphertext, and outputs μ only if the various checks pass. This allows us to achieve short ciphertexts, but still results in large public keys if using circuit obfuscation. To further compress the public key, we expect various techniques from the obfuscation literature to be useful. For example, the program P could take as input a succinct commitment to the set $\mathcal{J}$, together with a single decryption value d_i. It will output a decryption share associated with the set $\mathcal{J}$, and the share will be valid if and only if i is "turned on." Then outside of the obfuscated program, the decryption shares will be used to reconstruct the message the message μ, but reconstruction will only succeed if all the decryption shares are valid and all are associated with the same $\mathcal{J}$.

The Decoder Run-time Barrier. Now we turn to the more significant barrier. Namely, observe that Lemma 20 depends on the number of queries made by the decoder. In particular, Lemma 20 combined with Lemma 21 are invoked in a regime of parameters where the success probability is non-negligible but depends polynomially on the run-time of the decoder. Such statements are difficult to make with standard techniques in the literature. For example, an actual decoder may not work by explicitly querying the obfuscated code on various inputs, but may be able to extract information from a super-polynomial number of inputs in polynomial time. This would not contradict any statement that could be proved using iO together with standard cryptographic tools. But if the decoder could extract such information, it would invalidate our tracing guarantee.

A promising direction would be to use an exponentially-hard primitive, and scale the security parameter down to a logarithmic value depending on the decoder run-time. Exponential hardness used in this way allows for making the kind of fine-grained statements we would need for tracing; for an example see [Zha16]. Even using existing techniques involving exponential hardness, however, it seems non-trivial to prove our candidate secure, due to the structure of our tracing mechanism. Roughly, it seems that we would need an exponentially-secure obfuscation for the subset containment functionality, which can be obtained by techniques in [BKM+18, BLMZ19, GZ19]. Importantly, we need the obfuscation to be "optimally secure" in the sense that it is impossible to speed up finding an accepting input much faster than generic attacks. Unfortunately, the only constructions we have are proven in generic models, and also result in very big objects, growing with the universe size, which corresponds to the number of users.

We therefore leave as an interesting open question trying to adapt these ideas to get a provably-secure construction.

References

ABG+13. Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013.

ADVW13. Shweta Agrawal, Yevgeniy Dodis, Vinod Vaikuntanathan, and Daniel Wichs. On continual leakage of discrete log representations. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 401–420. Springer, Berlin, Heidelberg, December 2013.

AJS17. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation for Turing machines: Constant overhead and amortization. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 252–279. Springer, Cham, August 2017.

BBF+25. Rishiraj Bhattacharyya, Jan Bormet, Sebastian Faust, Pratyay Mukherjee, and Hussien Othman. CCA-secure traceable threshold (ID-based) encryption and application. Cryptology ePrint Archive, Report 2025/341, 2025.

BCP14. Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Berlin, Heidelberg, February 2014.

BF99. Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, Berlin, Heidelberg, August 1999.

BFO26. Jan Bormet, Sebastian Faust, and Hussien Othman. Threshold traitor tracing revisited: Insider attacks and multi-traitor tracing. Cryptology ePrint Archive, Paper 2026/422, 2026.

BGI+01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001.

BHO25. Jan Bormet, Jonas Hofmann, and Hussien Othman. Traceable threshold encryption without a trusted dealer. Cryptology ePrint Archive, Report 2025/342, 2025.

BKM+18. Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. A simple obfuscation scheme for pattern-matching with wildcards. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 731–752. Springer, Cham, August 2018.

BLMZ19. James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. New techniques for obfuscating conjunctions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 636–666. Springer, Cham, May 2019.

BN08. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 501–510. ACM Press, October 2008.

BP08. Olivier Billet and Duong Hieu Phan. Efficient traitor tracing from collusion secure codes. In Reihaneh Safavi-Naini, editor, *ICITS 08*, volume 5155 of *LNCS*, pages 171–182. Springer, Berlin, Heidelberg, August 2008.

BPR24a. Dan Boneh, Aditi Partap, and Lior Rotem. Accountability for misbehavior in threshold decryption via threshold traitor tracing. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 317–351. Springer, Cham, August 2024.

BPR24b. Dan Boneh, Aditi Partap, and Lior Rotem. Traceable secret sharing: Strong security and efficient constructions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 221–256. Springer, Cham, August 2024.

BPR25. Dan Boneh, Aditi Partap, and Lior Rotem. Traceable verifiable random functions. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part II*, volume 16001 of *LNCS*, pages 203–234. Springer, Cham, August 2025.

BS95. Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465. Springer, Berlin, Heidelberg, August 1995.

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Berlin, Heidelberg, May / June 2006.

BZ14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Berlin, Heidelberg, August 2014.

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Berlin, Heidelberg, August 1994.

CMMS25. Anirban Chakrabarti, Monosij Maitra, Arup Mondal, and Kushaz Sehgal. Silent threshold traitor tracing & enhancing mempool privacy. In *ACM CCS 2025*, pages 1769–1783. ACM Press, November 2025.

CPP25. Sébastien Canard, Nathan Papon, and Duong Hieu Phan. Public traceability in threshold decryption. Cryptology ePrint Archive, Report 2025/1347, 2025.

CVW+18. Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369. Springer, Cham, November 2018.

CW26. Jeffrey Champion and David J. Wu. Distributed monotone-policy encryption for DNFs from lattices. Cryptology ePrint Archive, Paper 2026/318, 2026.

DDP+25. Sourav Das, Pratish Datta, Aditi Partap, Swagata Sasmal, and Mark Zhandry. Optimal threshold traitor tracing. Cryptology ePrint Archive, Paper 2025/2154, 2025.

DPP20. Xuan Thanh Do, Duong Hieu Phan, and David Pointcheval. Traceable inner product functional encryption. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 564–585. Springer, Cham, February 2020.

FG25. Oriol Farràs and Miquel Guiot. Traceable secret sharing schemes for general access structures. Cryptology ePrint Archive, Report 2025/1120, 2025.

FG26. Oriol Farràs and Miquel Guiot. Information-theoretic strong traceable secret sharing schemes. Cryptology ePrint Archive, Paper 2026/435, 2026.

GJP25. Vipul Goyal, Abhishek Jain, and Aditi Partap. Traceable secret sharing revisited. Cryptology ePrint Archive, Report 2025/1980, 2025.

GKRW18. Rishab Goyal, Venkata Koppula, Andrew Russell, and Brent Waters. Risky traitor tracing and new differential privacy negative results. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 467–497. Springer, Cham, August 2018.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018.

GL89. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

GLW23. Junqing Gong, Ji Luo, and Hoeteck Wee. Traitor tracing with $N^{1/3}$-size ciphertexts and $O(1)$-size keys from k-Lin. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 637–668. Springer, Cham, April 2023.

GSS21. Vipul Goyal, Yifan Song, and Akshayaram Srinivasan. Traceable secret sharing and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 718–747, Virtual Event, August 2021. Springer, Cham.

GZ19. Steven D. Galbraith and Lukas Zobernig. Obfuscated fuzzy hamming distance and conjunctions from subset product problems. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 81–110. Springer, Cham, December 2019.

Hof24. Charlotte Hoffmann. Traceable secret sharing based on the chinese remainder theorem. Cryptology ePrint Archive, Report 2024/811, 2024.

JKL09. Pascal Junod, Alexandre Karlov, and Arjen K. Lenstra. Improving the Boneh-Franklin traitor tracing scheme. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 88–104. Springer, Berlin, Heidelberg, March 2009.

Juk01. Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. 01 2001.

KD98. Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 145–157. Springer, Berlin, Heidelberg, May / June 1998.

LW11. Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 568–588. Springer, Berlin, Heidelberg, May 2011.

NP98. Moni Naor and Benny Pinkas. Threshold traitor tracing. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 502–517. Springer, Berlin, Heidelberg, August 1998.

NP01. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, Berlin, Heidelberg, February 2001.

Tar03. Gábor Tardos. Optimal probabilistic fingerprint codes. In *35th ACM STOC*, pages 116–125. ACM Press, June 2003.

TS06. Dongvu Tonien and Reihaneh Safavi-Naini. An efficient single-key pirates tracing scheme using cover-free families. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS 2006*, volume 3989 of *LNCS*, pages 82–97. Springer, Berlin, Heidelberg, June 2006.

Zha16. Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 479–508. Springer, Berlin, Heidelberg, August 2016.

Zha20. Mark Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 652–682. Springer, Cham, August 2020.

Zha21. Mark Zhandry. White box traitor tracing. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 303–333, Virtual Event, August 2021. Springer, Cham.

Zha25. Mark Zhandry. Optimal traitor tracing from pairings. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part III*, volume 15603 of *LNCS*, pages 305–335. Springer, Cham, May 2025.