Traitor Tracing for Threshold Decryption

Dan Boneh, Aditi Partap, and Lior Rotem

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

Abstract. In a traitor tracing system there are n parties and each party holds a secret key. A broadcaster uses an encryption key to encrypt a message m to a ciphertext c so that every party can decrypt c using its secret key and obtain m. Suppose a subset of parties $\mathcal{J} \subseteq [n]$ combine their secret keys to create a pirate decoder $D(\cdot)$ that can decrypt ciphertexts from the broadcaster. Then it is possible to trace D to at least one member of $\mathcal{J}$ using only blackbox access to the decoder. Traitor tracing received much attention over the years and multiple schemes have been developed.

In this paper we explore how to do traitor tracing in the context of a threshold decryption scheme. Again, there are n parties and each party has a secret key, but now t parties are needed to decrypt a ciphertext c, for some $t > 1$. If a subset $\mathcal{J}$ of at least t parties use their secret keys to create a pirate decoder $D(\cdot)$, then it must be possible to trace D to at least one member of $\mathcal{J}$. This problem has not yet been explored in the literature, however, it has recently become quite important due to the use of encrypted mempools, as we explain in the paper.

We develop the theory of traitor tracing for threshold decryption. While there are several non-threshold traitor tracing schemes that we can leverage, adapting these constructions to the threshold decryption settings requires new cryptographic techniques. We present a number of constructions for traitor tracing for threshold decryption, and note that much work remains to explore the large design space.

1 Introduction

Accountability is needed in many applications of cryptography. In the context of digital signatures, accountability is often needed when using threshold signatures: if a quorum of t-out-of-n parties sign an invalid message, there is a need to identify the misbehaving quorum. An accountable threshold signature scheme is often built from a multisignature scheme [40,5,42,7,12,10,15] where every signature securely identifies the set of signing parties that generated it.

In this paper we develop the concept of accountability for threshold decryption. In a t-of-n threshold decryption scheme, there are n parties, and party i obtains at setup a *secret key share* sk_i. Later, when that party chooses to decrypt a ciphertext c, it uses sk_i to output a *decryption share*. Another party, which we call the *combiner*, collects t decryption shares and uses them to obtain a plaintext m, possibly using a public *combiner key* denoted by *pkc*.

The concern is that an adversary could offer to buy the secret key shares from some parties. Without accountability, this offer provides a risk-free profit for the parties. Our goal is to ensure that if some parties sell their secret key shares, then it is possible to securely identify those parties — or some subset of them — and hold them accountable. Here we are assuming that a whistleblower, William, reveals the information that the adversary purchased. William wants to ensure that this information securely identifies the guilty parties. We explain below why this is a good framing of the problem.

William might hope that reporting a set of well-formed secret key shares will identify the subset who sold their keys. However, this is clearly insecure. First, for common threshold decryption schemes (such as [30,52,23,9] and many others), a quorum of t parties has enough information to derive the secret key share of every other party. This lets them frame an innocent quorum by giving the adversary key shares that belong to some other parties. Second, only a naive decryption party would give the adversary its decryption key share as is. A more sophisticated set of parties would jointly generate a malformed key that lets the adversary decrypt, but cannot be traced to anyone.

We will need to defend against both issues raised above. The second issue suggests that a sophisticated set of parties will not sell the adversary a key. Instead, they will jointly construct an "obfuscated" decoder algorithm D that takes as input a ciphertext c and outputs a plaintext m. The challenge then is to identify the guilty set of parties using only black-box access to this decoder D. This question is closely related to traitor tracing [25], but in a very different context. Traitor tracing schemes are designed for settings where every party can decrypt a ciphertext on its own. Think of a set of DVD players, where every player needs to decrypt an encrypted DVD disk on its own. In our settings, at least t secret key shares are needed to decrypt a ciphertext. As such, this work generalizes the traitor tracing problem to the settings of threshold decryption. We consider two very different situations.

Case 1: A decoder from a greater-than-threshold quorum. Suppose that a coalition of f traitors, where $f \geq t$, constructs a decoder algorithm $D(\cdot)$. They have enough information to construct a decoder that takes as input a ciphertext c and outputs its decryption. Our goal is to trace this decoder to at least one member of the coalition of traitors, possibly using a tracing key tk generated at setup. We give precise definitions in Section 3.

A natural starting point to construct such a system is any of the (non-threshold) traitor tracing schemes that are fully collusion resistant. Some such systems are built from pairings [16,32,56,55,33], some are built from lattices [34,24], and some are built from iO [21], to name a few. We explain in Section 3 that adapting these schemes to the threshold settings requires new techniques. In this paper we look at using a fully collusion-resistant scheme due to Boneh and Naor [14] (see Section 4). This traitor tracing scheme has *constant* size ciphertext and public key. Secret key shares in this scheme are large; however, for the application described below, this is quite reasonable. Adapting this non-threshold traitor tracing system to the threshold decryption settings requires a new cryptographic primitive we call *Bipartite Threshold Encryption* (technically, Bipartite Threshold KEM), as described in Section 4.1. We give three constructions for a BTKEM: a direct (but inefficient) combinatorial construction, an efficient construction with short ciphertexts from DDH, and a further improvement providing short public keys using pairings. These constructions illustrate the added complexity in constructing traitor tracing systems for threshold decryption.

In an upcoming work, we also show how to adapt the recent traitor tracing scheme of Gong, Lou, and Wee [33] to the threshold setting using very different techniques.

Case 2: A decoder from a below-threshold quorum. We next turn to tracing a decoder D built from a coalition $\mathcal{J}$ of f traitors, where $f < t$. This decoder cannot decrypt a ciphertext on its own and needs to take additional decryption shares as input. In particular, for a ciphertext c the decoder D is invoked as $D(c, d_1, \ldots, d_k)$, where $d_1, \ldots, d_k$ are decryption shares for c from some set of parties $\mathcal{S} \subseteq [n]$ of size $|\mathcal{S}| = k \geq t - f$. We consider two types of decoders:

- A *universal decoder* outputs the correct decryption of a well-formed ciphertext c (with non-negligible probability) whenever $|\mathcal{J} \cup \mathcal{S}| \geq t$. That is, the decoder will decrypt a well-formed c whenever it has enough information to do so. In Section 6 we describe a generic tracing procedure for such a decoder, assuming that the threshold decryption scheme is semantically secure. The tracing procedure will output at least one member of the coalition $\mathcal{J}$ that created D.
- An *exact decoder* outputs the correct decryption of a well-formed ciphertext c (with non-negligible probability) only when $|\mathcal{S}| = t - f$ and $|\mathcal{J} \cup \mathcal{S}| = t$. This is a more restrictive

decoder that only takes $t - f$ valid decryption shares as input. Tracing is not possible in this case because, without knowledge of $\mathcal{J}$, the tracing algorithm cannot find a set $\mathcal{S}$ such that $|\mathcal{S}| = t - f$ and $|\mathcal{J} \cup \mathcal{S}| = t$. To see why, observe that the number of such sets is only $\binom{n-f}{t-f}$, and this can be a negligible fraction of the total number of subsets of size $t - f$. For example, when $t = n/3$ and $f = t/2$, the fraction is $2^{-\Omega(n)}$. Consequently, we show in Section 6 that for a robust threshold decryption scheme, the tracing algorithm can never get the decoder to work (the decoder always outputs $\perp$), and this implies that tracing is impossible.

Instead, we design in Section 6 a *confirmation* algorithm. The algorithm takes as input a suspect coalition $\mathcal{J}^*$ and uses blackbox access to the decoder D to convince a verifier that $\mathcal{J}^*$ is the traitor set.

Note that we disallow a decoder D that only works for a constant number of specific sets $\mathcal{S}^*$ where $|\mathcal{J} \cup \mathcal{S}^*| = t$. In this extreme case, neither confirmation nor tracing is possible, by a similar argument as above.

1.1 Motivation

Why study traitor tracing for threshold decryption? First, accountability for threshold decryption is a well-motivated question that is interesting in its own right. It comes up naturally in many applications of threshold decryption, such as voting [26] and auctions [50]. In all these applications one wants accountability for parties who sell their secret key shares. Second, this topic has recently become important due to the introduction of encrypted mempools [4] using threshold decryption in several blockchain projects [44,28]. The goal of an encrypted mempool is to keep the data in a block secret until the block is finalized (posted) on chain. After finalization, the block should become available in the clear so that all the transactions in the block can be executed. Keeping transaction data hidden until the block is finalized is intended to prevent a certain value extraction technique called MEV [27,2,46]. We refer to Rondelet and Kilbourn [47] for a detailed analysis of the benefits and limitations of deploying an encrypted mempool. Several encrypted mempool designs use identical consensus and decryption committees [1,4]. The idea is that when a validator signs a block, it also releases one decryption share for that block. This way, once two thirds of the validators in the consensus committee sign the block (thereby finalizing it), there are also enough decryption shares to decrypt the block (the decryption threshold is set to two thirds of the committee).

A critical weakness of this design is that an adversary could bribe members of the consensus committee into creating a decoder D that lets the adversary decrypt blocks *before* they are finalized. This lets the adversary engage in MEV extraction and defeats the purpose of encrypting the mempool. Crucially, without the ability to trace the decoder D, this bribe provides a risk-free profit for the validators, so there is no reason for them to refuse the bribe. Traitor tracing for threshold decryption puts the validators at risk of being slashed and may compel them to not engage with the adversary.

Why focus on tracing a decoder. So far we focused on tracing a stateless decoder to at least one member of the coalition that created that decoder. Consequently, a sophisticated set of t or more traitors who want to evade tracing, might refuse to give the adversary a decoder. Instead, they might jointly offer an anonymous decryption service: the adversary sends a well-formed target ciphertext c to the service, the traitors jointly decrypt it, and send back (anonymously) the decryption of c. This will defeat our tracing algorithms because the set of traitors can maintain state across different decryption requests. For example, if they ever receive a request to decrypt a malformed ciphertext (which is how tracing often works) they could panic and shut down the decryption service, refusing to answer any more requests. This will defeat tracing algorithms designed for stateless decoders, although other, more complex, tracing strategies may apply [35,36,53,45].

The decryption service option, however, may be unappealing to an adversary trying to decrypt a ciphertext for its own benefit. The parties running the decryption service will learn the plaintext before the adversary. They could then use the plaintext for their own benefit and cut off the adversary. In other words, for the application at hand, the adversary may only be willing to pay the bribe in exchange for a decoding algorithm that the adversary can run on its own.

1.2 Additional related work

The literature on traitor tracing is vast. Some schemes, such as [25,38,11], are designed to trace a pirate decoder that was created by a coalition of bounded size. Schemes that are secure against an arbitrary coalition size are said to be *fully collusion resistant*.

The first fully collision-resistant traitor tracing scheme with a public key, ciphertext, and secret key that are all sublinear in the number n of parties was proposed by Boneh, Sahai, and Waters [16] (see also [20]). Their scheme was based on an assumption over composite order bilinear groups, and enjoyed public key and ciphertext of size $O(\sqrt{n})$ and constant size secret keys. Garg, Kumarasubramanian, Sahai, and Waters [32] subsequently achieved similar parameters using prime order bilinear groups. In [55], Wee constructed a functional encryption scheme for quadratic functions, that can be used to reproduce the result of [32] directly using the framework of [16]. In a recent breakthrough result, Zhandry [56] constructed a pairing-based traitor tracing scheme where the public key, ciphertext, and secret key were all of size $O(n^{1/3})$. His proof of security was in the generic group model [51]. This was later on improved by Gong, Luo, and Wee [33], who got the secret key size down to a constant while relying on standard assumptions over bilinear groups.

There are also traitor-tracing schemes from lattice-based assumptions. The groundbreaking work of Goyal, Koppula, and Waters [34] constructed the first fully collision-resistant traitor tracing scheme directly from lattice-based assumptions. Their construction boasts essentially optimal parameters, where the public key, ciphertext size, and secret keys, all grow poly-logarithmically in n. Their construction was later greatly simplified by Chen et al. [24]. In Section 3 we discuss the techniques underlying all these pairing-based and lattice-based schemes in more detail.

Optimal traitor tracing can also be achieved from indistinguishability obfuscation [21]. We believe that this construction lends itself to the threshold decryption setting in a fairly straightforward manner.

Naor and Pinkas studied [41] *threshold traitor tracing*, where the goal is to trace a pirate decoder that successfully decrypts an input ciphertext with some threshold probability. This is unrelated to traitor tracing for threshold decryption.

The recent work of Li et al. [39] also considered accountability for decryption. However, their focus was completely dissimilar to ours, as they considered very different notions of accountability in the non-threshold setting and hardware-based solutions.

2 Preliminaries

Threshold Decryption Schemes. We begin by defining threshold decryption schemes [29,30]. The definitions we use follow those of Boneh and Shoup [19]. A threshold decryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine})$ is a tuple of four polynomial-time algorithms:

1. $\mathsf{KeyGen}(1^\lambda, n, t) \to (pk, pkc, sk_1, \ldots, sk_n)$ is the probabilistic **key generation algorithm**. It takes in the security parameter $\lambda \in \mathbb{N}$, the number n of decryptors and the threshold t. It outputs a public key pk, a combiner public key pkc, and secret keys $sk_1, \ldots, sk_n$.
2. $\mathsf{Enc}(pk, m) \to c$ is the probabilistic **encryption algorithm**. It takes as input the public key pk and a message m, and it outputs a ciphertext c.
3. $\mathsf{Dec}(sk_i, c) \to d_i$ is the deterministic **decryption algorithm**. It takes in a decryptor's secret key sk_i and a ciphertext c, and its output is a decryption share d_i of c under sk_i.
4. $\mathsf{Combine}(pkc, c, \mathcal{J}, \{d_j\}_{j\in\mathcal{J}}) \to m/\bot$ is the deterministic **combiner algorithm**. Its input is a combiner public key pkc, a ciphertext c, a subset $\mathcal{J}$ of $[n]$, and decryption shares $\{d_j\}_{j\in\mathcal{J}}$. It outputs either a message m or a rejection symbol $\bot$.

Correctness. An honestly generated ciphertext should be correctly decrypted by any subset of t decryptors. Specifically, for all security parameters $\lambda \in \mathbb{N}$, all $n, t \in \mathbb{N}$, all t-size subsets $\mathcal{J} \subseteq [n]$, and all messages m in the message space, it holds that

$$\Pr\left[m' = m \ : \ \begin{array}{c} (pk, pkc, sk_1, \ldots, sk_n) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t) \\ c \xleftarrow{\$} \mathsf{Enc}(pk, m) \\ \forall i \in \mathcal{J},\ d_i \leftarrow \mathsf{Dec}(sk_i, c) \\ m' \leftarrow \mathsf{Combine}(pkc, c, \mathcal{J}, \{d_j\}_{j\in\mathcal{J}}) \end{array} \right] = 1.$$

Security. We require that a threshold decryption scheme satisfies the standard notion of **semantic security**. Many applications require the stronger notion of security against chosen-ciphertext attacks (CCA security). We do not consider CCA security in this paper since our focus is the notion of traitor tracing, which seems orthogonal to CCA security. The notion of semantic security is captured by the security game in Fig. 1.

Definition 1 (Semantic security). *We say that $\mathcal{E}$ satisfies* semantic security *if for every probabilistic polynomial time adversary $\mathcal{A}$, the following function is negligible in λ:*

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

Robustness. Most scenarios call for a *robust* threshold decryption scheme. By that, we mean that a decryption share d_i can be publicly verified, to make sure that it is indeed a valid decryption share of a ciphertext c, originating from party i. Syntactically, the KeyGen algorithm now outputs an additional verification key vk and the Dec algorithm outputs a robustness proof π_i along with the decryption share d_i. We define an additional algorithm, $\mathsf{ShareVf}(pk, vk, c, (d_i, \pi_i), i)$, that takes as input the public key pk, the verification key vk, a cipher text c and a decryption share along with the corresponding robustness proof (d_i, π_i) from party i, and the index i of that party. It outputs 0 or 1, denoting whether (d_i, π_i) is a valid decryption share for c for party i. For correctness, we require that $\mathsf{ShareVf}$ outputs 1 for an honestly generated share, i.e. $\mathsf{ShareVf}(pk, vk, c, (d_i, \pi_i), i) = 1$ for any $(pk, pkc, sk_1, \ldots, sk_n) \leftarrow\$ \mathsf{KeyGen}(1^\lambda, n, t), c \xleftarrow{\$} \mathsf{Enc}(pk, m), (d_i, \pi_i) \leftarrow \mathsf{Dec}(sk_i, c)$ for all values of n, t, m, i. We may sometimes implicitly assume that in a robust scheme, each decryption share d_i encodes the party i from which it originated. For security, we require that the threshold decryption scheme satisfies **decryption consistency**. Informally, it means that an adversary cannot produce two different shares $d_i, \hat{d}_i$ with valid robustness proofs for any party i, and any ciphertext c, even if it is given the secret keys for all parties. This is captured by the security game in Fig. 2.

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

$\mathcal{J} \leftarrow \emptyset$
$(n, t, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
$(pk, pkc, sk_1, \dots, sk_n) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t)$
$(m_0, m_1) \xleftarrow{\$} \mathcal{A}^{\mathsf{corrupt}(\cdot)}(\mathsf{state}, pk, pkc)$
$b \xleftarrow{\$} \{0,1\},\ c \xleftarrow{\$} \mathsf{Enc}(pk, m_b)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{corrupt}(\cdot)}(\mathsf{state}, c)$
if $|\mathcal{J}| \geq t$ **then return** 0
if $b' = b$ **then return** 1 **else return** 0

Oracle $\mathsf{corrupt}(i)$

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

Fig. 1. The semantic security experiment for a threshold decryption scheme $\mathcal{E}$ and an adversary $\mathcal{A}$.

Definition 2 (Robustness). *We say that $\mathcal{E}$ satisfies robustness if, for every probabilistic polynomial time adversary $\mathcal{A}$, the following function is negligible in λ:*

$$\mathsf{Adv}^{\mathrm{dc}}_{\mathcal{A},\mathcal{E}}(\lambda) := \Pr\left[\mathbf{DC}_{\mathcal{A},\mathcal{E}}(\lambda) = 1\right]$$

Experiment $\mathbf{DC}_{\mathcal{A},\mathcal{E}}(\lambda)$

$\mathcal{J} \leftarrow \emptyset$
$(n, t, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
$(pk, pkc, sk_1, \dots, sk_n, vk) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t)$
$(c, (d_i, \pi_i), (\hat{d}_i, \hat{\pi}_i), i) \xleftarrow{\$} \mathcal{A}(\mathsf{state}, pk, pkc, sk_1, \dots, sk_n, vk)$
if $d_i \neq \hat{d}_i \wedge \mathsf{ShareVf}(pk, vk, c, (d_i, \pi_i), i) = \mathsf{ShareVf}(pk, vk, c, (\hat{d}_i, \hat{\pi}_i), i) = 1$ **then**
  return 1
else return 0

Fig. 2. The security game $\mathbf{DC}_{\mathcal{A},\mathcal{E}}(\lambda)$ capturing robustness for a threshold decryption scheme $\mathcal{E}$ and an adversary $\mathcal{A}$.

Looking ahead, our constructions in Sections 4 and 5 do not explicitly provide robustness (since this is not the focus of this work), but can be made robust using standard techniques. One of our constructions in Section 6 relies on a generic threshold decryption scheme, that is robust as per Definition 2 above.

KEM vs. encryption. In certain parts of the paper, we may find it convenient to work with *threshold key-encapsulation mechanisms (KEMs)* rather than threshold decryption schemes. Syntactically, a threshold KEM is defined by the same algorithms as a threshold decryption scheme, but the **encapsulation algorithm** Enc gets only the public key pk (and no message) as input. It outputs a ciphertext c but also a key k from some key space $\mathcal{K} = \mathcal{K}(\lambda)$ induced by the KEM. Accordingly, $\mathsf{Combine}$ outputs a key k' from the key space, rather than a message. Correctness requires that for all $\lambda \in \mathbb{N}$, all $0 < t < n$, and all t-size subsets $\mathcal{J}$ of $[n]$ it holds that

$$\Pr\left[k' = k \;\middle|\; \begin{array}{c} (pk, pkc, sk_1, \ldots, sk_n) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t) \\ (c, k) \xleftarrow{\$} \mathsf{Enc}(pk) \\ d_i \leftarrow \mathsf{Dec}(sk_i, c) \text{ for } i \in \mathcal{J} \\ k' \leftarrow \mathsf{Combine}(pkc, c, \mathcal{J}, \{d_i\}_{i \in \mathcal{J}}) \end{array} \right] = 1.$$

Semantic security is defined similarly to that of threshold decryption with the following changes to the $\mathbf{IND\text{-}CPA}_{\mathcal{A},\mathcal{E}}(\lambda)$ experiment: the adversary $\mathcal{A}$ now does not choose messages (m_0, m_1). Instead, Enc outputs a key k together with c. If $b = 1$, then when $\mathcal{A}$ is given the ciphertext c, it is also given the key k. If $b = 0$, then $\mathcal{A}$ is given a uniformly random key k' in the key space $\mathcal{K}$.

3 Tracing Large Traitor Coalitions

In this section, we define traitor tracing for threshold KEM schemes, in the setting where the number f of corruptions is greater than the threshold t.

A traitor tracing threshold KEM (TTT-KEM for short) is a threshold KEM (recall Section 2) with the following modifications:

- $\mathsf{KeyGen}(1^\lambda, n, t, 1^{1/\epsilon}) \to (pk, pkc, sk_1, \ldots, sk_n, \boldsymbol{tk})$ now also takes in an error parameter $\epsilon = \epsilon(\lambda)$ and outputs an additional tracing key tk.
- There is an additional PPT **tracing algorithm** Trace that is invoked as $\mathcal{J} \xleftarrow{\$} \mathsf{Trace}^{D(\cdot)}(pk, tk, 1^{1/\epsilon})$. The algorithm takes as input the public key pk, the tracing key tk, and an error bound $\epsilon = \epsilon(\lambda)$. Trace also has oracle access to a *"decoder"* algorithm D. It outputs a subset $\mathcal{J} \subseteq [n]$.

Informally, if the decoder D, given a ciphertext c, can learn any information about the encapsulated key, then Trace traces D back to a party out of the coalition that "manufactured" it. The following definition defines a secure traitor tracing threshold KEM using the security experiment in Fig. 3.

Experiment $\mathbf{ExpTrace}_{\mathcal{A},\mathcal{E},\epsilon}(\lambda)$	
1 :	$\mathcal{J} \leftarrow \emptyset$
2 :	$(n, t, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
3 :	$(pk, pkc, sk_1, \ldots, sk_n, tk) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t, 1^{1/\epsilon(\lambda)})$
4 :	$D \xleftarrow{\$} \mathcal{A}^{\mathsf{corrupt}(\cdot)}(\mathsf{state}, pk, pkc)$ // output a decoder alg. D
5 :	$\mathcal{J}' \xleftarrow{\$} \mathsf{Trace}^{D(\cdot)}(pk, tk, 1^{1/\epsilon(\lambda)})$ // trace decoder
6 :	**return** $(pk, D, \mathcal{J}, \mathcal{J}')$

Fig. 3. The tracing experiment for a threshold KEM $\mathcal{E}$ and an adversary $\mathcal{A}$. The corruption oracle $\mathsf{corrupt}(\cdot)$ is defined as in Fig. 1.

Definition 3. *Let $\epsilon = \epsilon(\lambda)$ and $\delta = \delta(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A traitor-tracing threshold KEM $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine}, \mathsf{Trace})$ with key space $\mathcal{K} = \{\mathcal{K}_\lambda\}$ is (ϵ, δ)-secure if for every* PPT *adversary $\mathcal{A}$ and for every $\lambda \in \mathbb{N}$, it holds that*

1. $\Pr[\mathsf{GoodTr} = 1] \geq \Pr[\mathsf{GoodDec} = 1] - \delta(\lambda)$.
2. $\Pr[\mathsf{BadTr} = 1] \leq \delta(\lambda)$.

where $(pk, D, \mathcal{J}, \mathcal{J}') \xleftarrow{\$} \mathbf{ExpTrace}_{\mathcal{A},\mathcal{E},\epsilon}(\lambda)$ as defined in Figure 3. The events GoodDec*,* GoodTr*, and* BadTr *are defined as follows:*

- GoodDec *occurs when $P(D) \geq 1/2 + \epsilon(\lambda)$, where $P(D)$ is defined by*

$$P(D) = \Pr\left[D(c, k_b) = b \ : \ (c, k_0) \xleftarrow{\$} \mathsf{Enc}(pk), k_1 \xleftarrow{\$} \mathcal{K}(\lambda), b \xleftarrow{\$} \{0,1\}\right]$$

  Note that for a fixed D, $P(D)$ is a number in $[0,1]$, and it is a random variable over the choice of D.
- GoodTr *holds when $\mathcal{J}' \neq \emptyset$ and $\mathcal{J}' \subseteq \mathcal{J}$.*
- BadTr *holds when $\mathcal{J}' \neq \emptyset$ and $\mathcal{J}' \not\subseteq \mathcal{J}$.*

We say that $\mathcal{E}$ is secure *if there exists a negligible function $\nu = \nu(\lambda)$ such that $\mathcal{E}$ is $(1/p, \nu)$-secure for every polynomial $p = p(\lambda)$. For an adversary $\mathcal{A}$ and a TTT-KEM scheme $\mathcal{E}$, we denote*

$$\mathsf{Adv}^{\mathrm{tt}}_{\mathcal{A}_1,\mathcal{TTT}\mathcal{E}}(\lambda) := \max\left\{\Pr[\mathsf{GoodDec} = 1] - \Pr[\mathsf{GoodTr} = 1], \Pr[\mathsf{BadTr} = 1]\right\}.$$

Semantic security and robustness for TTT-KEM schemes are defined as in Section 2, but the adversary is also given the tracing key tk in both experiments. In case the TTT-KEM is robust, then in the **ExpTrace**, the adversary is also given the share verification key vk.

On "thresholdizing" traitor tracing schemes. Definition 3 is a strict generalization of standard traitor tracing schemes for non-threshold KEM or public key encryption. In particular, setting the threshold $t = 1$ gives the traitor tracing definition for non-threshold schemes. This suggests that TTT-KEMs are harder to construct than standard traitor tracing schemes.

Existing approaches for traitor tracing. Given the above, a natural avenue to realize the notion of TTT-KEM is to try and "thresholdize" existing traitor tracing schemes. We briefly survey some prominent schemes that might serve as the basis for TTT-KEM constructions:[1] Starting with the work of Boneh, Sahai, and Waters [16] (BSW hereinafter), most efficient constructions of traitor tracing go through variants of a notion called **Private Linear Broadcast Encryption (PLBE)**. Since the work of BSW, there have been many works constructing pairing-based PLBE or variants thereof [20,32,55], constructions of PLBE from lattices [34,24], and a construction from indistinguishability obfuscation [21]. Zhandry [56] recently constructed a pairing-based traitor tracing scheme that deviates from the PLBE framework considerably, but still relies on the general idea of a broadcast system with private revocation.

A completely different approach for traitor tracing was suggested by Boneh and Naor [14] (see also [6]) and utilizes **fingerprinting codes** [17]. Using such codes, they showed how one can construct a traitor tracing scheme with constant-size ciphertexts from any public key encryption scheme. Zhandry [56] observed that by replacing the underlying public-key encryption scheme with an identity-based encryption (IBE) scheme, the public-key size is reduced to constant size.

[1] We focus here on schemes that offer *full* collusion resistance, as in our definition. Many other constructions focus on defending against more restricted traitor coalitions whose size is a-priori bounded.

From traitor tracing to TTT-KEM. Converting the above schemes to TTT-KEM is harder than might first appear. The traditional route of converting a cryptosystem into a threshold variant of itself is by secret-sharing some secret key material among all parties. This approach seems to be at odds with the traitor tracing task: in such transformations, a coalition of $f \geq t$ corrupted parties can typically reconstruct the secret key of all other parties, and hence can also frame any innocent party. More generally, to allow tracing, traitor-tracing schemes need to guarantee some sort of (computational) independence among the parties' keys. This independence needs to be preserved even given more than t secret keys, which seems to go against the traditional thresholdizing via secret-sharing paradigm. Different traitor-tracing schemes employ different algebraic or combinatorial mechanisms to ensure this independence among keys, and so different insights may be necessary in order to convert them into TTT-KEMs. In this paper, we initiate this endeavor by constructing a TTT-KEM starting from the work of Boneh and Naor [14]. We believe that future work may do the same for other traitor tracing schemes, namely PLBE-based schemes from standard assumptions, relying on different techniques.

Tracing vs. semantic security. In the context of *non-threshold* traitor tracing, Zhandry [56] recently observed that any scheme that provides tracing vis-à-vis a definition similar to Definition 3, automatically provides semantic security as well. In other words, an adversary that breaks semantic security can be used to break the tracing guarantee. Consider a successful adversary breaking the semantic security of a KEM with an advantage of at least 2ϵ for some non-negligible ϵ. Then the state of this adversary after observing the public key pk can be seen as a decoder. With non-negligible probability, this decoder will have an advantage of at least ϵ in distinguishing real keys from random, and so the event $\mathsf{GoodDec}$ occurs with non-negligible probability. However, the semantic security adversary has corrupted no one! Hence, the event GoodTr occurs with probability 0. This breaks the tracing definition.

In the threshold setting, however, this is no longer the case. The reason is that the semantic security adversary *can* corrupt up to $t-1$ parties. Hence, GoodTr can occur with strictly positive probability. As a toy example, say that we have a traitor-tracing threshold KEM $\mathcal{E}$ that satisfies both tracing security and semantic security. Consider a new (flawed) scheme $\mathcal{E}'$ that on threshold t instantiates $\mathcal{E}$ on threshold $t' < t$. $\mathcal{E}'$ inherits the tracing security of $\mathcal{E}$, but it is not semantically secure: an adversary in possession of $t' < t$ keys can decrypt any ciphertext. This shows that for threshold KEM, tracing security does not imply semantic security.

Knowing ϵ in advance. Note that our definition requires that KeyGen gets a lower bound on the decoder's success probability ϵ. Naor and Pinkas [41] refer to such schemes as "threshold traitor tracing" schemes (not to be confused with our notion of TTT-KEM). The Boneh and Naor traitor tracing scheme is a threshold traitor tracing scheme in this sense, and hence Our construction in Section 4, which builds on their ideas, will also share the same property. For scenarios in which this assumption may be unwanted, Zhandry [56] presented a general compiler that transforms any threshold traitor tracing scheme to a non-threshold one, without changing the dependency of the scheme's parameters on the number n of parties. His compilation carries over to our setting.

4 Traitor Tracing from Bipartite Threshold KEM

In this section, we construct a traitor tracing threshold KEM scheme. We begin with a brief overview.

Overview of the construction. The starting point of our construction is the standard traitor tracing PKE scheme of Boneh and Naor [14]. Their construction relies on a primitive called *fingerprinting codes* [18,54,14]. A (binary) fingerprinting code is a set of words $\Gamma = \{\bar{w}^{(1)}, \ldots, \bar{w}^{(n)}\}$, where each $\bar{w}^{(j)}$ is an ℓ-bit string. 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 $W \subseteq \Gamma$ of words, and it comes up with a word $\bar{w}$. It can decide on $\bar{w}$ as it pleases but there is a stipulation: $\bar{w}$ has to be constructable by mixing and matching the bits of the words in W. That is, if for some index i, the ith bit of all words in W is 0, then the ith bit of $\bar{w}$ must be 0 as well (and similarly if the ith bit of all words is 1). Call such a word *feasible* for W. The guarantee of fingerprinting codes is that Trace can trace $\bar{w}$ back to least one of the words in W.

Equipped with such a code Γ, Boneh and Naor constructed a traitor tracing scheme from any public key encryption scheme $\mathcal{E}$. The idea is to generate 2ℓ key pairs for $\mathcal{E}$, where ℓ is the length of the words in Γ. The public key consists of all 2ℓ public keys, and can be seen as a matrix with ℓ rows and two columns in which jth row is $(pk_{j,0}, pk_{j,1})$ for $j = 1, \ldots, \ell$. Denote the secret key associated with these public keys by $(sk_{j,0}, sk_{j,1})$. To decide on the secret keys, Boneh and Naor associate the ith party with the ith word in the code, $\bar{w}^{(i)}$. The secret key of party i is then $(sk_{1,w_1^{(i)}}, \ldots, sk_{\ell,w_\ell^{(i)}})$ where $\bar{w}^{(i)} = w_1^{(i)} \ldots w_\ell^{(i)}$. To encrypt a message m, one chooses a random index $j \xleftarrow{\$} [\ell]$, and encrypts m under both $pk_{j,0}$ and $pk_{j,1}$ to obtain to ciphertexts c_0 and c_1. The final ciphertext is $c = (j, c_0, c_1)$. Note that any party with a secret key can decrypt either c_0 or c_1.

Tracing relies on the observation that, thanks to the semantic security of $\mathcal{E}$, a coalition of parties holding only $sk_{j,0}$ 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 m'. The same goes for a coalition holding only $sk_{j,1}$. This observation can be used to extract a word $\bar{w}^*(D)$ from a pirate decoder D with the property that $\bar{w}^*(D)$ is 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.

Now let us try and extend the approach of Boneh and Naor to the threshold setting. A naive attempt would be to replace the use of the public key encryption $\mathcal{E}$ with a threshold decrpytion scheme. Immediately, we can see that this approach runs into a correctness issue. The problem is this: say that a coalition of t parties wants to decrypt some ciphertext $c = (j, c_0, c_1)$. It may very well be the case that some parties can compute decryption shares for c_0 and others can compute decryption shares for c_1. These decryption shares do not match and hence do not allow the decryption of c.

It turns out that what we need is a new kind of threshold decryption that we call "bipartite threshold decryption". Loosely speaking, we can think of bipartite threshold decryption as having two-sided ciphertexts (c_0, c_1), and each party is given a secret key that can decrypt either c_0 or c_1. Moreover, it should satisfy two seemingly contradicting requirements:

- **"One side security:"** An adversary that holds only keys that can partially decrypt c_0 should not be able to 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 m' (an analogous condition should hold if the adversary can only partially decrypt c_1). This property, which we call *one side security*, is necessary for the tracing argument to go through and it is satisfied by the naive construction above. Informally speaking, one side security implies that c_0 and c_1 must be **computationally independent** from the standpoint of such an adversary.

– **"Two sides correctness:"** Still, any coalition holding t secret keys must be able to decrypt c to recover the encrypted message. This should hold regardless of how many of these keys can decrypt c_0 and how many can decrypt c_1. This condition, which we call *two sides correctness*, is necessary for correctness to hold, and it implies that conditioned on the public key, c_0 and c_1 **must be correlated** somehow. This condition *is not met* by the naive construction.

We formally define bipartite encryption in Section 4.1, where for convenience, we focus on KEM rather than PKE. The formal definition addresses several technical aspects that are ignored in this informal overview. In section 4.2 we construct a traitor tracing scheme for threshold KEM from such a bipartite threshold KEM and fingerprinting codes and prove its security. In our case, the correlation that must be introduced between the two parts of the ciphertext introduces some subtleties to the security proofs that were not covered in this overview.

In Section 5 we show that even though our requirements of bipartite threshold KEM might seem contradictory at first glance, they *can* be constructed and even efficiently. First, we formally define the building blocks for our traitor tracing scheme.

4.1 Building Blocks

Our construction of TTT-KEM relies on two building blocks: a collusion resistant fingerprinting code [18,54] and a new notion we call a bipartite threshold KEM. We explain each in turn.

Collusion Resistant Fingerprinting Codes Originally, collusion-resistant fingerprinting codes (or fingerprinting codes for short) were introduced for fingerprinting digital content [18,54] but they have found applications to traitor tracing as well (e.g., [25,37,14]). In order to present fingerprinting codes, we first 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 trinary 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 $F_\delta(W) = \{\bar{w} \in F(W) \ : \ \bar{w} \text{ is } \delta\text{-noisy}\}$.

Formally, a **fingerprinting code** is a pair $\mathcal{C} = (\mathsf{FCGen}, \mathsf{FCTrace})$ of algorithms:

– $\mathsf{FCGen}(1^n, \epsilon, \delta) \to (\Gamma, tk)$ is the probabilistic **code generation algorithm**. It takes in a number $n > 0$ of words, a security parameter ϵ, and a noise bound δ. It outputs a code Γ, which is a set of n words in $\{0,1\}^\ell$, and a tracing key tk. The length ℓ of the code words is a function $\ell = \ell(n, \epsilon, \delta)$ of the code size n, the security parameter ϵ, and the noise bound δ.
– $\mathsf{FCTrace}(\bar{w}^*, tk) \to S$ is the deterministic **tracing algorithm**. It takes as input a noisy word $\bar{w}^*$ and the tracing key tk, and outputs a subset S of $[n]$.

The security property of fingerprinting codes says that if an adversary in possession of a subset $W \subseteq \Gamma$ of codewords generates a δ-noisy word $\bar{w}^*$ which is feasible for W (that is, $\bar{w}^* \in F_\delta(W)$), then $\mathsf{FCTrace}(\bar{w}^*, tk)$ outputs a set S that corresponds to a non-empty subset of W. This is formally captured by the security game in Fig. 4, which is parameterized by n, ϵ, δ, and the subset $\mathcal{I}$ of indices that defines the subset W of words given to the adversary.

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

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

Fig. 4. The security experiment for a fingerprinting code $\mathcal{C}$ 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.

Definition 4. *A fingerprinting code $\mathcal{C}$ 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*

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

Known constructions. The fingerprinting codes we use, that allow for tracing even in the presence of a bounded fraction of unknown bits (represented by '?' symbols), are called *robust* fingerprinting codes. These were originally introduced by Boneh and Naor [14], who constructed the first such codes based on Boneh-Shaw codes [17]. Their construction achieved codewords of length $O(n^4 \log(n/\epsilon) \log(1/\epsilon)/(1-\delta)^2)$, where n is the number of words in the code, ϵ is the tracing error, and δ is an upper bound on the fraction of '?' in the noisy word that is given to the tracing algorithm. This was later improved by Nuida [43] and by Boneh, Kiayias, and Montgomery [13]. The latter work is based on Tardos codes [54] and enjoys codewords of length $O((n \log n)^2 \log(n/\epsilon)/(1-\delta))$. This is optimal up to logarithmic factors.

Bipartite threshold KEM The second building block that we will use is a new notion that we put forth, called a **bipartite threshold KEM** scheme, or a BT-KEM scheme for short. A BT-KEM scheme is a threshold KEM scheme, where instead of one set of secret keys, there are 2ℓ sets, for a parameter ℓ of the scheme. We think of these keys as arranged in 2ℓ rows, or "positions", and in each position, there are two possible secret keys for each party: a *left* key and a *right* key. Similarly, a ciphertext is now comprised of two parts: a left ciphertext and a right ciphertext. A BT-KEM should satisfy two properties: one side security and two sides correctness as we discuss above.

Formally, A BT-KEM is a tuple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine})$ of algorithms with the following syntax:

– $\mathsf{KeyGen}(1^\lambda, n, t, \ell) \to (pk, pkc, \{(sk^{(j)}_{1,0}, sk^{(j)}_{1,1}), \ldots, (sk^{(j)}_{n,0}, sk^{(j)}_{n,1})\}_{j \in [\ell]})$ 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 a number ℓ of "positions". It outputs a public key pk, a combiner public key pkc, and n secret keys $sk_1, \ldots, sk_n$. Each secret key is made up of 2ℓ keys: $sk_i = \{sk_{i,0}^{(j)}, sk_{i,1}^{(j)}\}_{j\in[\ell]}$ for $i = 1, \ldots, n$. $sk_{i,0}^{(j)}$ is called the left secret key of i at position j and $sk_{i,1}$ is called the right secret key of i at position j.

- $\mathsf{Enc}(pk, j) \to (c_0, c_1, k)$ is the probabilistic **encapsulation algorithm**. It takes as input the public key pk and a position $j \in [\ell]$, and it outputs a ciphertext $c = (c_0, c_1)$ and a key k. c_0 is called the left ciphertext and c_1 is called the right ciphertext.
- $\mathsf{Dec}(j, sk_{i,b}^{(j)}, c = (c_0, c_1)) \to d_i$ is the deterministic **decryption algorithm**. It takes in a decryptor's secret key $sk_{i,b}$ — where $i \in [n]$, $j \in [\ell]$, and $b \in \{0, 1\}$ — and a ciphertext c, and its output is a decryption share d_i of c under $sk_{i,b}^{(j)}$. To simplify notation, we may assume $sk_{i,b}^{(j)}$ encodes the bit b even if this is not explicitly noted.
- $\mathsf{Combine}(pkc, j, c, \mathcal{J}, \{d_i\}_{i\in\mathcal{J}}) \to k/\perp$ is the deterministic **combiner algorithm**. Its input is a combiner public key pkc, a position $j \in [\ell]$, a ciphertext c, a subset $\mathcal{J}$ of $[n]$, and decryption shares $\{d_i\}_{i\in\mathcal{J}}$. It outputs either a key k or a rejection symbol $\perp$.

Definition 5 (correctness). *A BT-KEM scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine})$ *is said to be correct if for every polynomially-bounded functions* $n = n(\lambda)$, $t = t(\lambda)$, *and* $\ell = \ell(\lambda)$ *of the security parameter* $\lambda \in \mathbb{N}$ *the following holds:*
For all $\lambda \in \mathbb{N}$, *positions* $j \in [\ell]$, *subsets* $\mathcal{J} \subseteq [n]$ *of size* $t(\lambda)$, *and* n*-bit strings* $s \in \{0, 1\}^n$ *it holds that*

$$\Pr\left[k = k' \ : \ \begin{array}{c} (pk, pkc, \{(sk_{1,0}^{(j)}, sk_{1,1}^{(j)}), \ldots, (sk_{n,0}^{(j)}, sk_{n,1}^{(j)})\}_{j\in[\ell]}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t) \\ (c, k) \xleftarrow{\$} \mathsf{Enc}(pk, j) \\ d_i \leftarrow \mathsf{Dec}(j, sk_{i,s_i}^{(j)}, c) \ \textit{for}\ i \in \mathcal{J} \\ k' \leftarrow \mathsf{Combine}(pkc, j, c, \mathcal{J}, \{d_i\}_{i\in\mathcal{J}}) \end{array} \right] = 1.$$

Experiment $\mathbf{ExpBTKEM}_{\mathcal{A},\mathcal{E}}(\lambda)$

1 : $(n, t, \ell, u, d, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(1^\lambda)$
2 : $(pk, pkc, \{(sk_{1,0}^{(j)}, sk_{1,1}^{(j)}), \ldots, (sk_{n,0}^{(j)}, sk_{n,1}^{(j)})\}_{j\in[\ell]}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t, \ell)$
3 : $(c^{(0)} = (c_0^{(0)}, c_1^{(0)}), k^{(0)}) \xleftarrow{\$} \mathsf{Enc}(pk, u)$
4 : $(c^{(1)} = (c_0^{(1)}, c_1^{(1)}), k^{(1)}) \xleftarrow{\$} \mathsf{Enc}(pk, u)$
5 : $c_d^* \leftarrow c_d^{(d)}$
6 : $b \xleftarrow{\$} \{0, 1\}$
7 : **if** $b = 0$ **then** $c_{1-d}^* \leftarrow c_{1-d}^{(1-d)}$ **else** $c_{1-d}^* \leftarrow c_{1-d}^{(d)}$
8 : $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, pk, pkc, (c_0^*, c_1^*), k^{(0)}, \{(sk_{1,0}^{(j)}, sk_{1,1}^{(j)}), \ldots, (sk_{n,0}^{(j)}, sk_{n,1}^{(j)})\}_{j\in[\ell]\setminus\{u\}}, (sk_{1,d}^{(u)}, \ldots, sk_{n,d}^{(u)}))$
9 : **return** $(b' = b)$

Fig. 5. The one side security experiment for a BT-KEM $\mathcal{E}$ and an adversary $\mathcal{A}$.

Semantic security. We require that a BT-KEM scheme satisfies semantic security. Observe that any BT-KEM scheme can be seen as a standard threshold KEM scheme with additional syntactic properties, and so semantic security is already defined in Section 2, with the following modifications to the **IND-CPA** security experiment:

– In addition to n and t, the adversary $\mathcal{A}$ also chooses the number ℓ of positions.
– The secret key of party i is made up of all of its 2ℓ secret keys – the left key and the right key of each position. That is,

$$sk_i = ((sk_{i,0}^{(1)}, sk_{i,1}^{(1)}), \ldots, (sk_{i,0}^{(\ell)}, sk_{i,1}^{(\ell)})).$$

– The challenge ciphertext is computed by the challenger by invoking the encapsulation algorithm with respect to a random position in $[\ell]$. That is, the challenger first samples a uniformly random $j \xleftarrow{\$} [\ell]$, and then samples $(c = (c_0, c_1), k) \xleftarrow{\$} \mathsf{Enc}(pk, j)$.

One side security. In addition, we require that a BT-KEM scheme satisfies "one side security" as discussed at the beginning of this section. The security game for one side security is given in Fig. 5 and the security notion is formally captured by the following definition.

Definition 6. *We say that $\mathcal{E}$ has* one side security *if for every probabilistic polynomial time adversary $\mathcal{A}$, the following function is negligible in λ:*

$$\mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A},\mathcal{E}}(\lambda) := \left|2\Pr[\mathbf{ExpBTKEM}_{\mathcal{A},\mathcal{E}}(\lambda) = 1] - 1\right| .$$

4.2 Our Construction of TTT-KEM

Equipped with the above two building blocks, we are now ready to present our construction of a TTT-KEM.

Let $\mathcal{BTE} = (\mathcal{BTE}.\mathsf{KeyGen}, \mathcal{BTE}.\mathsf{Enc}, \mathcal{BTE}.\mathsf{Dec}, \mathcal{BTE}.\mathsf{Combine})$ be a bipartite threshold decryption scheme and let $\mathcal{C} = (\mathcal{C}.\mathsf{FCGen}, \mathcal{C}.\mathsf{FCTrace})$ be a collusion-resistant fingerprinting code. Assume for simplicity that the keys outputted by $\mathcal{BTE}$ are uniformly distributed over its key space $\mathcal{K}$.[2] Our construction uses the following subroutine, TR that takes in a public key pk, an index $j \in [\ell]$, an integer B, and three bits $b_k, b_0, b_1 \in \{0,1\}$, and has oracle access to a decoder D. The subroutine TR^D is defined as follows:

1. Set $ctr \leftarrow 0$.
2. For $r = 1, \ldots, B$:
   (a) Sample $(c^{(0)} = (c_0^{(0)}, c_1^{(0)}), k^{(0)}) \xleftarrow{\$} \mathcal{BTE}.\mathsf{Enc}(pk, j)$ and $(c^{(1)} = (c_0^{(1)}, c_1^{(1)}), k^{(1)}) \xleftarrow{\$} \mathcal{BTE}.\mathsf{Enc}(pk, j)$.
   (b) Set $c^* \leftarrow (j, c_0^{(b_0)}, c_1^{(b_1)})$.
   (c) Query D on $(c^*, k^{(b_k)})$. If the response is 1, set $ctr \leftarrow ctr + 1$.
3. Return ctr.

Equipped with the above building blocks, our TTT-KEM scheme, denoted $\mathcal{TTTE}$, is defined in Fig. 6.

It is immediate that construction is correct and semantically secure, by the correctness and semantic security of the underlying BT-KEM. The following theorem establishes that it also provides tracing.

[2] This assumption will be satisfied by our constructions of BT-KEM schemes. Generally speaking, note that given a BT-KEM scheme not satisfying the assumption, we can always construct a scheme for which this condition *is* satisfied by applying a randomness extractor to the key.

A TTT-KEM scheme $\mathcal{TTTE}$

$\mathcal{TTTE}$.KeyGen$(1^\lambda, n, t, 1^{1/\epsilon})$:

1. Set $\delta \leftarrow (1/2 - \epsilon)/(1/2 - 2/\sqrt{\lambda})$ and sample a fingerprinting code: $(\mathit{\Gamma}, tk) \xleftarrow{\$} \mathcal{C}.\mathsf{FCGen}(1^n, 2^{-\lambda}, \delta)$.
2. Sample keys for $\mathcal{BTE}$: $(pk, pkc, \{(sk^{(j)}_{1,0}, sk^{(j)}_{1,1}), \ldots, (sk^{(j)}_{n,0}, sk^{(j)}_{n,1})\}_{j \in [\ell]}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t, \ell)$.
   // $\ell = \ell(n, \lambda, \epsilon)$ is the words length of the code $\mathit{\Gamma}$
3. Set $pk' \leftarrow pk$, $pkc' \leftarrow pkc$ and $tk' \leftarrow tk$.
4. For $i = 1, \ldots, n$, set $sk'_i \leftarrow (sk^{(1)}_{i, w^{(i)}_1}, \ldots, sk^{(\ell)}_{i, w^{(i)}_\ell})$
   // $\bar{w}^{(i)}$ is the ith word in the code $\mathit{\Gamma}$
5. Return $(pk', pkc', sk'_1, \ldots, sk'_n, tk')$.

$\mathcal{TTTE}$.Enc(pk):

1. Sample $j \xleftarrow{\$} [\ell]$.
2. $(c_0, c_1, k) \xleftarrow{\$} \mathcal{BTE}.\mathsf{Enc}(pk, j)$.
3. Let $c = (j, c_0, c_1)$ and return (c, k).

$\mathcal{TTTE}$.Dec(sk_i, c):

1. Parse sk_i as $(sk^{(1)}_{i, w^{(i)}_1}, \ldots, sk^{(\ell)}_{i, w^{(i)}_\ell})$ and c as (j, c_0, c_1).
2. Compute $d_i \leftarrow \mathcal{BTE}.\mathsf{Dec}(j, sk^{(j)}_{i, w^{(i)}_j}, (c_0, c_1))$.
3. Return d_i.

$\mathcal{TTTE}$.Combine$(pkc, c, \mathcal{J}, \{d_i\}_{i \in \mathcal{J}})$:

1. Parse c as (j, c_0, c_1) and set $c' \leftarrow (c_0, c_1)$.
2. Compute $k \leftarrow \mathcal{BTE}.\mathsf{Combine}(pkc, j, c', \mathcal{J}, \{d_i\}_{i \in \mathcal{J}})$.
3. Return k.

$\mathcal{TTTE}$.Trace$^D(pk, tk, 1^{1/\epsilon})$:

1. Set $N \leftarrow \lambda^2$ and $B \leftarrow \lambda^{3/2}$.
2. For $j = 1, \ldots, \ell$:
   (a) Compute $p_{001} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 0, 0, 1)$, $p_{100} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 1, 0, 0)$, and $p_{111} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 1, 1, 1)$.
   (b) Set $a_0 \leftarrow |p_{001} - p_{100}|$ and $a_1 \leftarrow |p_{001} - p_{111}|$.
   (c) If $a_0 \geq B$, set $w_j \leftarrow 0$. If not, check if $a_1 \geq B$, and if so set $w_j \leftarrow 1$. If both conditions do not hold, set $w_j \leftarrow$ '?'.
3. Run $\mathcal{J} \leftarrow \mathsf{FCGen}.\mathsf{Trace}(tk, \bar{w}^*)$, where $\bar{w}^* \leftarrow w_1 w_2 \ldots w_\ell$.
4. Return $\mathcal{J}$.

Fig. 6. Our TTT-KEM scheme, denoted $\mathcal{TTTE}$, built from a fingerprinting code and a BT-KEM.

Theorem 1. *For every probabilistic polynomial time algorithm $\mathcal{A}_1$ there exists a probablistic polynomial-time algorithm $\mathcal{A}_2$ such that*

$$\mathsf{Adv}^{\mathrm{tt}}_{\mathcal{A}_1,\mathcal{TTTE}}(\lambda) \le 9\ell \cdot e^{-\lambda/24} + 2\ell\lambda^2 \cdot \mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A}_2,\mathcal{BTE}}(\lambda)$$

for all $\lambda \in \mathbb{N}$, where $\ell = \ell(\lambda, n, \epsilon)$ is the codeword length as defined in the scheme.

We stress that our definition of decoders is much more general than the one originally considered by Boneh and Naor [14]. Whereas they considered tracing decoders that *fully decrypt* with high probability, we consider (in line with more modern definition of traitor tracing) tracing any decoder that can learn any non-trivial information regarding the plaintext (via a semantic security type definition). This on its own poses new technical challenges that require new technical insight in the security proof.

Proof. Let $\mathbf{Exp_0}$ denote the original **ExpTrace** experiment and let $\mathcal{A}_1$ be an adversary participating in the experiment. We will show that the probability that the word $\bar{w}^*$ computed by Trace in the $\mathbf{Exp_0}$ experiment is not feasible for the set $\mathcal{J}$ of corrupted parties is small. Concretely, let $\bar{w}^*(\mathcal{A}_1)$ be the random variable describing the word $\bar{w}^*$ constructed by Trace in $\mathbf{Exp_0}$. We also define J to be the random variable describing the set $\mathcal{J}$ of corrupted parties in the experiment, and C to be the random variable corresponding to the code generated by $\mathcal{C}.\mathsf{FCGen}$. Let $W(C, J) = \{\bar{w}^{(i)}\}_{i \in J}$, where $C = \{\bar{w}^{(1)}, \dots, \bar{w}^{(n)}\}$, and denote $F(J) = F(W(C, J))$.

The following lemma bounds the probability that $\bar{w}'(\mathcal{A})$ is not δ-noisy.

Lemma 1. *Suppose that*

$$\Pr\left[D(c, k_b) = b \ : \ (c, k_0) \xleftarrow{\$} \mathcal{TTTE}.\mathsf{Enc}(pk), k_1 \xleftarrow{\$} \mathcal{K}(\lambda), b \xleftarrow{\$} \{0,1\}\right] \ge \frac{1}{2} + \epsilon.$$

Then, it holds that

$$\Pr\left[\bar{w}^*(\mathcal{A}_1) \text{ is not } \delta\text{-noisy}\right] \le 4\ell \cdot e^{-\lambda/24}.$$

Proof. We first prove that D must be a good distinguisher for a $(1-\delta)$-fraction of the positions $j \in [\ell]$. Concretely, say that a position $j \in [\ell]$ is *good* if

$$\Pr\left[D(j, c, k_b) = b \ : \ \begin{array}{l} (c, k_0) \xleftarrow{\$} \mathcal{BTE}.\mathsf{Enc}(pk, j) \\ k_1 \xleftarrow{\$} \mathcal{K}(\lambda), b \xleftarrow{\$} \{0,1\} \end{array}\right] \ge \frac{1}{2} + \frac{2}{\sqrt{\lambda}}. \quad (1)$$

We argue that at least $(1-\delta)$-fraction of the positions $j \in [\ell]$ are good. Suppose that this is not the case. Then, it holds that

$$\begin{aligned} &\Pr\left[D(c, k_b) = b \ : \ (c, k_0) \xleftarrow{\$} \mathcal{TTTE}.\mathsf{Enc}(pk), k_1 \xleftarrow{\$} \mathcal{K}(\lambda), b \xleftarrow{\$} \{0,1\}\right] \\ &\quad < \delta \cdot \left(\frac{1}{2} + \frac{2}{\sqrt{\lambda}}\right) + (1-\delta) \cdot 1 \\ &\quad = \delta \cdot \left(\frac{2}{\sqrt{\lambda}} - \frac{1}{2}\right) + 1 \\ &\quad = \frac{\frac{1}{2} - \epsilon}{\frac{1}{2} - \frac{2}{\sqrt{\lambda}}} \cdot \left(\frac{2}{\sqrt{\lambda}} - \frac{1}{2}\right) + 1 \\ &\quad = \frac{1}{2} + \epsilon. \end{aligned}$$

This stands in contradiction to the assumption in the claim, and so at least $(1-\delta)$-fraction of the positions $j \in [\ell]$ are good.

We now show that for all good positions j, it holds that the jth symbol in $\bar{w}'(\mathcal{A})$ is '?' with negligible probability. Let $j \in [\ell]$ be a good position. Since j is good, this means that in particular

$$\left|\Pr\left[D((j, c_0^{(0)}, c_1^{(0)}), k^{(1)}) = 1\right] - \Pr\left[D((j, c_0^{(1)}, c_1^{(1)}), k^{(1)}) = 1\right]\right| \geq \frac{4}{\sqrt{\lambda}},$$

where $(c^{(b)} = (c_0^{(b)}, c_1^{(b)}), k^{(b)}) \xleftarrow{\$} \mathcal{BTE}.\mathsf{Enc}(pk, j)$ for $b \in \{0,1\}$. By the triangle inequality, this means that at least one of the following inequalities holds:

$$\left|\Pr\left[D((j, c_0^{(0)}, c_1^{(0)}), k^{(1)}) = 1\right] - \Pr\left[D((j, c_0^{(1)}, c_1^{(0)}), k^{(1)}) = 1\right]\right| \geq \frac{2}{\sqrt{\lambda}}, \tag{2}$$

or

$$\left|\Pr\left[D((j, c_0^{(1)}, c_1^{(1)}), k^{(1)}) = 1\right] - \Pr\left[D((j, c_0^{(1)}, c_1^{(0)}), k^{(1)}) = 1\right]\right| \geq \frac{2}{\sqrt{\lambda}}. \tag{3}$$

Consider first the case where Eq. (2) holds, and assume without loss of generality that

$$\Pr\left[D((j, c_0^{(0)}, c_1^{(0)}), k^{(1)}) = 1\right] > \Pr\left[D((j, c_0^{(1)}, c_1^{(0)}), k^{(1)}) = 1\right]$$

(the analysis in the complementing case is symmetric). This means that p_{001} is distributed as a binomial random variable with N trials and probability parameter η for some $\eta \in [0,1]$, and p_{100} is distributed binomially with N trials and probability parameter $\eta' \geq \eta + \frac{2}{\sqrt{\lambda}}$. In particular, the expected values of p_{100} and p are at least $2N/\sqrt{\lambda} = 2\lambda^{3/2}$ far apart.

Recall that Trace sets $w_j = 0$ if $a_0 = |p_{001} - p_{100}| \geq B = \lambda^{3/2}$. Hence, it will set $w_j = 0$ in the specific case where p_{100} and p_{001} are both at most $N/2\sqrt{\lambda}$ far away from their expected values. Hence

$$\begin{aligned}
\Pr\left[w_j \neq 0\right] &= \Pr\left[a_0 < N^{3/2}\right] \\
&\leq \Pr\left[\left(p_{100} < \mathbb{E}[p_{100}] - \frac{N}{2\sqrt{\lambda}}\right) \vee \left(p_{001} > \mathbb{E}[p_{001}] + \frac{N}{2\sqrt{\lambda}}\right)\right] \\
&\leq \Pr\left[p_{100} < \mathbb{E}[p_{100}] - \frac{N}{2\sqrt{\lambda}}\right] + \Pr\left[p_{001} > \mathbb{E}[p_{001}] + \frac{N}{2\sqrt{\lambda}}\right] \\
&\leq 2 \cdot \Pr_{X \xleftarrow{\$} \mathsf{Bin}(N, 1/2)}\left[|X - \mathbb{E}[X]| > \frac{N}{2\sqrt{\lambda}}\right],
\end{aligned} \tag{4}$$

where X is sampled from the binomial distribution with N trials and probability parameter $1/2$ and Eq. (4) follows from convexity. By the Chernoff bound, it holds that

$$\Pr\left[w'_j \neq 0\right] \leq 4e^{-\lambda/24}. \tag{5}$$

Now suppose Eq. (2) does not hold. Then, this means that Eq. (3) holds and a similar analysis shows that $a_1 \geq B$ with probability at least $1 - 4e^{-\lambda/24}$. Hence, in thise case, Trace sets $w_j \neq$ '?' with probability at least $1 - 4e^{-\lambda/24}$. The proof of the lemma is concluded by taking a union bound over all good positions and noting that there at most ℓ such positions. ∎

The following lemma bounds the probability that the word computed by Trace is not feasible for the set of words induced by the adversary's corruption queries.

Lemma 2. *There exists a probablistic polynomial-time algorithm $\mathcal{A}_2$ such that*

$$\Pr\left[\bar{w}^*(\mathcal{A}_1) \notin F(J)\right] \leq 4\ell \cdot e^{-\lambda/6} + 2\ell\lambda^2 \cdot \mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A}_2,\mathcal{BTE}}(\lambda).$$

Proof. Consider the experiment $\mathbf{Exp_1}$ obtained from $\mathbf{Exp_0}$ by the following modification: instead of running the true Trace algorithm, the challenger runs a $\mathsf{Trace}'_{\mathcal{J}}$ algorithm that depends on the set $\mathcal{J}$ of parties that the adversary corrupted before outputting the decoder D. The modified tracing algorithm is defined by:

1. Set $N \leftarrow \lambda^2$ and $B \leftarrow N^{3/2}$.
2. For $j = 1, \ldots, \ell$:
   (a) If $w_j^{(i)} = 0$ for all $i \in \mathcal{J}$ then compute $p \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 0, 0, 0)$. Otherwise, compute $p \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 0, 1, 1)$.
   (b) Compute $p_{100} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 1, 0, 0)$, and $p_{111} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 1, 1, 1)$.
   (c) Set $a_0 \leftarrow |p - p_{100}|$ and $a_1 \leftarrow |p - p_{111}|$.
   (d) If $a_0 \geq B$, set $w_j \leftarrow 0$. If not, check if $a_1 \geq B$, and if so set $w_j \leftarrow 1$. If both conditions do not hold, set $w_j \leftarrow$ '?'.
3. Run $\mathcal{J} \leftarrow \mathsf{FCGen.Trace}(tk, \bar{w}^*)$, where $\bar{w}^* \leftarrow w_1 w_2 \ldots w_\ell$.

We abuse notation and let C and J be defined as before but over $\mathbf{Exp_1}$ (observe that they are identically distributed in both experiments). Let $\bar{w}'(\mathcal{A}_1)$ be the random variable describing the word $\bar{w}'$ constructed by Trace'_J in $\mathbf{Exp_1}$. We wish to claim that $\bar{w}'(\mathcal{A}_1) \in F(J)$ with high probability.

Claim 2 $\Pr\left[\bar{w}'(\mathcal{A}_1) \notin F(J)\right] \leq 4\ell \cdot e^{-\lambda/6}$.

Proof. We will prove that

$$\Pr\left[\bar{w}'(\mathcal{A}_1) \notin F(J)\right] \leq 4\ell \cdot e^{-B^2/6N}. \tag{6}$$

and the claim will follow by our choice of N and B. Let $j \in [\ell]$ and consider two cases:

1. Say that $w_j^{(i)} = 1$ for all $i \in \mathcal{J}$. We wish to upper bound the probability that Trace'_J sets $w'_j = 0$. Since $w_j^{(i)} = 1$, p is computed in the jth iteration of $\mathsf{Trace}'_{\mathcal{J}}$ by $p \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 0, 1, 1)$. Moreover, $\mathsf{Trace}'_{\mathcal{J}}$ sets $w'_j = 0$ only if $a_0 = |p - p_{100}| \geq B$ in the j iteration, where $p_{100} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 1, 0, 0)$. Note that p_{100} and p are identically distributed. Hence, it holds that

$$\begin{aligned} \Pr\left[a_0 \geq B\right] &= \Pr\left[|p - p_{100}| \geq B\right] \\ &\leq \Pr\left[\left|p - \frac{N}{2}\right| \geq \frac{B}{2} \vee \left|p_{100} - \frac{N}{2}\right| \geq \frac{B}{2}\right] \\ &\leq 2 \cdot \Pr\left[\left|p - \frac{N}{2}\right| \geq \frac{B}{2}\right]. \end{aligned} \tag{7}$$

   Observe that p sampled from the binomial distribution with N trials and some probability parameter $\eta \in [0, 1]$. By convexity, the probability in eq. (7) is maximized when $\eta = 1/2$. Hence, by the Chernoff bound, it holds that

$$\Pr\left[a_0 \geq B\right] \leq 4e^{-B^2/6N}.$$

2. Say that $w_j^{(i)} = 0$ for all $i \in \mathcal{J}$. Then, the probability that Trace'_J sets $w'_j = 1$ is bounded by $4e^{-B^2/6N}$. The analysis is symmetrical to the previous case, noting that in this case, p is computed as the outputs of $\mathsf{TR}^D(pk, j, N, 0, 0, 0)$, which is identically distributed to p_{111}, and that w'_j is set to 1 only if $a_1 = |p - p_{111}|$ is at least B.

We call an index $j \in [\ell]$ *fixed* for $\mathcal{J}$ if for all $i_1, i_2 \in \mathcal{J}$ it holds that $w_j^{(i_1)} = w_j^{(i_2)}$; that is, the jth bit of all code words corresponding to $\mathcal{J}$ are the same. The above analysis shows that for all indices $j \in [\ell]$ that are fixed for $\mathcal{J}$, $\mathsf{Trace}'_{\mathcal{J}}$ computes a false w'_j with probability at most $4e^{-B^2/6N}$. Eq. (6) then follows by a union bound over all indices $j \in [\ell]$. This concludes the proof of the claim. □

We now relate the probability that $\bar{w}'(\mathcal{A}_1) \notin F(J)$ to the probability that $\bar{w}^*(\mathcal{A}_1) \notin F(J)$. Note that for the event $\bar{w}^*(\mathcal{A}_1) \notin F(J)$ to occur, it must be the case that for some position $j \in [\ell]$ that is fixed for J, the jth character $\bar{w}^*(\mathcal{A}_1)$ is inconsistent with $W(C, J)$; that is, if all words corresponding to corrupted parties have 0 as their jth bit, then $\bar{w}^*(\mathcal{A}_1)$ has 1 in its jth entry, or vice versa. However, if all words corresponding to corrupted parties have 0 as their jth bit, then $\mathcal{A}_1$ has only left keys in position j. Hence, the one side security of $\mathcal{BTE}$ implies that it should not be able to distinguish between a ciphertext of the form $(j, c_0^{(0)}, c_1^{(1)})$ as given to it by the real Trace algorithm in $\mathbf{Exp_0}$ and a ciphertext of the form $(j, c_0^{(0)}, c_1^{(0)})$ as given to it by $\mathsf{Trace}'_{\mathcal{J}}$ in $\mathbf{Exp_1}$. The lemma then follows by our bound of the probability that $\mathsf{Trace}'_{\mathcal{J}}$ sets w'_j to be 1 (incorrectly) in $\mathbf{Exp_1}$.

In more detail, we present an adversary for a generalized version of one-side security. Instead of one key-ciphertext pair, the adversary receives $B = \lambda^2$ such pairs $(k_1, c_1 = (c_{1,0}, c_{1,1}), \ldots, (k_B, c_B = (c_{B,0}, c_{B,1}))$, and it has to distinguish between the case where $c_{1,1-d}, \ldots, c_{B,1-d}$ are consistent with $k_1, \ldots, k_B$ and the case where these are independent random ciphertext halves. A standard hybrid argument shows that an adversary $\mathcal{A}_2$ with advantage ϵ' in this experiment can be transformed into an adversary $\mathcal{A}'_2$ with advantage ϵ/B for the original $\mathbf{ExpBTKEM}_{\mathcal{A}'_2,\mathcal{E}}(\lambda)$ experiment.

Consider an adversary $\mathcal{A}_2$ which invokes $\mathcal{A}_1(1^\lambda)$ and gets back n and t. It then samples a code $(\Gamma, tk) \xleftarrow{\$} \mathcal{C}.\mathsf{FCGen}(1^n, \epsilon, \delta)$, an index $u \xleftarrow{\$} [\ell]$, and a side $d \xleftarrow{\$} \{0,1\}$, and outputs (n, t, ℓ, u, d) as its outputs in Step 1 of the generalized $\mathbf{ExpBTKEM}_{\mathcal{A}_2,\mathcal{E}}(\lambda)$ experiment. In response, it gets B key-ciphertext pairs $(k_1, c_1 = (c_{1,0}, c_{1,1}), \ldots, (k_B, c_B = (c_{B,0}, c_{B,1}))$ and all secret keys but those that correspond to side $1-d$ at position u. $\mathcal{A}_2$ uses these keys and the code Γ to respond to corruption queries issued by $\mathcal{A}_1$. If at any point, $\mathcal{A}_1$ issues a query for which $\mathcal{A}_2$ does not posses the entire corresponding secret key (i.e., a query for a party i for which $w_u^{(i)} = 1-d$), then $\mathcal{A}_2$ outputs a random $b' \xleftarrow{\$} \{0,1\}$. Denote this event by guess. Otherwise, $\mathcal{A}_2$ outputs a decoder D. $\mathcal{A}_1$ then decides on its outputs as follows:

1. It computes p by invoking D on k_r for $r = 1, \ldots, B$ and sets p to be the number of times D returns 1.
2. It computes $p_{100} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 1, 0, 0)$, and $p_{111} \xleftarrow{\$} \mathsf{TR}^D(pk, j, N, 1, 1, 1)$.
3. $\mathcal{A}_1$ sets $a_0 \leftarrow |p_{001} - p_{100}|$ and $a_1 \leftarrow |p_{001} - p_{111}|$.
4. If $a_0 \geq B$, set $\mathcal{A}_1$ outputs 0. If not, its checks if $a_1 \geq B$, and if so it outputs 1. If both conditions do not hold, it outputs $\perp$.

Let fix denote the event over $\mathbf{Exp_0}$ in which there is at least one index that is fixed for J. fix can similarly be defined over $\mathbf{Exp_1}$ and has equal probability over both experiments. In the subsequent analysis, we assume that $\Pr[\mathsf{fix}] = 1$. This is without loss of generality: it may only increase the difference

$$\left|\Pr\left[\bar{w}^*(\mathcal{A}_1) \notin F(J)\right] - \Pr\left[\bar{w}'(\mathcal{A}_1) \notin F(J)\right]\right|$$

since if no index is fixed for J, then all words are feasible. In particualr,

$$\Pr\left[\bar{w}^*(\mathcal{A}_1) \notin F(J) \;\middle|\; \overline{\mathsf{fix}}\right] = \Pr\left[\bar{w}'(\mathcal{A}_1) \notin F(J) \;\middle|\; \overline{\mathsf{fix}}\right] = 0.$$

Since $\Pr[\mathsf{fix}] = 1$, it holds that $\Pr[\overline{\mathsf{guess}}] \geq 1/2\ell$ (note that $\Pr[\mathsf{guess}]$ is also identical over both experiments). We condition the rest of the analysis on $\overline{\mathsf{guess}}$. Suppose for the bit b chosen by the challenge in $\mathbf{ExpBTKEM}_{\mathcal{A}_2,\mathcal{E}}(\lambda)$ is equal to 1, then the ciphertext c^* is a valid ciphertext, distributed as the ciphertexts given to the decoder D in $\mathbf{Exp_1}$. Hence, the output of $\mathcal{A}_2$ is distributed identically to $w^*(\mathcal{A}_1)_u$. Similarly, if $b = 0$, the output of $\mathcal{A}_2$ is distributed identically to $w'(\mathcal{A}_1)_u$. It follows that the advantage of $\mathcal{A}_2$ in the generalized security experiment is

$$\frac{1}{2\ell} \cdot \left|\Pr\left[\bar{w}^*(\mathcal{A}_1) \notin F(J)\right] - \Pr\left[\bar{w}'(\mathcal{A}_1) \notin F(J)\right]\right|.$$

Hence, there is a probabilistic polynomial time adversary $\mathcal{A}_2'$ for the original experiment with advantage

$$\mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A}_2,\mathcal{BTE}}(\lambda) \geq \frac{1}{2\ell B} \cdot \left|\Pr\left[\bar{w}^*(\mathcal{A}_1) \notin F(J)\right] - \Pr\left[\bar{w}'(\mathcal{A}_1) \notin F(J)\right]\right|. \tag{8}$$

The lemma follows by incorporating Eq. (8) and (6).

Together, Lemmas 2 and 1 show that

$$\Pr\left[\bar{w}^*(\mathcal{A}_1) \notin F_\delta(J)\right] < 8\ell \cdot 2^{-\lambda/24} + 2\ell\lambda^2 \cdot \mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A}_2,\mathcal{BTE}}(\lambda). \tag{9}$$

To conclude the proof of the theorem, observe that conditioned on $\bar{w}^*(\mathcal{A}_1) \in F(J)$, Trace outputs an index in J with probability at least $1 - 2^{-\lambda}$ by the collusion resistance of the fingerprinting code $\mathcal{C}$.

5 Constructing Bipartite Threshold KEM

In this section, we construct BT-KEM schemes from various assumptions: the stronger the assumption, the more efficient the BT-KEM construction is, and hence also the resulting TTT-KEM scheme.

Before turning to our efficient constructions, we note that one can fairly easily construct a BT-KEM scheme from any threshold KEM scheme. The construction is rather inefficient, since the keys and the ciphertext scale linearly with the threshold t. We sketch this construction in Appendix A.

5.1 Constant-Size Ciphertexts from DDH

Although the parameters achieved by the generic construction are non-trivial, it still suffers from long ciphertexts. In this section, we show how to construct a BT-KEM scheme with constant-size ciphertexts by relying on the DDH assumption in cyclic groups. For asymptotic reasoning, we consider a distribution ensemble over such groups. This is formalized by the existence of a group generation algorithm $\mathcal{G}$ that takes the security parameter as input and outputs a triple $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a description of a group of order p generated by g. The scheme is formally defined in Fig. 7.

A BT-KEM scheme $\mathcal{BTDDH}$

Public parameters: $(\mathbb{G}, g, p)$ sampled by $\mathcal{G}(1^\lambda)$.

$\mathcal{BTDDH}$.KeyGen$(1^\lambda, n, t, \ell)$:

1. For $j = 1, \ldots, \ell$ do:
   (a) Sample $x_j, y_j, z_j \xleftarrow{\$} \mathbb{Z}_p$ and compute $X_j \leftarrow g^{x_j}$, $Y_j \leftarrow g^{y_j}$ and $Z_j \leftarrow g^{z_j}$.
   (b) Sample a Shamir t-out-of-n secret sharing $s_{j,1}, \ldots, s_{j,n}$ of x_j.
   (c) Set $pk_j \leftarrow (X_j, Y_j, Z_j)$, $sk_{i,0}^{(j)} \leftarrow s_{j,i}/y_j$ and $sk_{i,1}^{(j)} \leftarrow s_{j,i}/z_j$.
2. Set $pk \leftarrow (pk_1, \ldots, pk_\ell)$ and $pkc \leftarrow \perp$.
3. Return $(pk, pkc, \{(sk_{1,0}^{(j)}, sk_{1,1}^{(j)}), \ldots, (sk_{n,0}^{(j)}, sk_{n,1}^{(j)})\}_{j \in [\ell]})$.

$\mathcal{BTDDH}$.Enc(pk, j):

1. Parse pk as $(pk_1, \ldots, pk_\ell)$ and pk_j as (X_j, Y_j, Z_j).
2. Sample $r \xleftarrow{\$} \mathbb{Z}_p$ and set $k \leftarrow X_j^r$
3. Compute $c_0 \leftarrow Y_j^r$ and $c_1 \leftarrow Z_j^r$ and set $c \leftarrow (c_0, c_1)$
4. Return (k, c).

$\mathcal{BTDDH}$.Dec$(j, sk_{i,b}^{(j)}, c = (c_0, c_1))$:

1. Compute $d_i \leftarrow c_b^{sk_{i,b}^{(j)}}$.
2. Return d_i.

$\mathcal{BTDDH}$.Combine$(pkc, j, c, \mathcal{J}, \{d_i\}_{i \in \mathcal{J}})$:

1. Compute the Lagrange coefficients $(\lambda_i^{\mathcal{J}})_{i \in \mathcal{J}}$ corresponding to the subset $\mathcal{J}$:
   $\lambda_i^{\mathcal{J}} \leftarrow \prod_{v \in \mathcal{J} \setminus \{i\}} \frac{v}{v-i} \in \mathbb{Z}_p$.
2. Compute $k \leftarrow \prod_{i \in \mathcal{J}} d_i^{\lambda_i^{\mathcal{J}}}$.
3. Return k.

Fig. 7. Our BT-KEM in cyclic groups, denoted $\mathcal{BTDDH}$.

Correctness. Let $0 < t < n$ and let $\ell > 0$. Let $\mathcal{J}$ be a subset of size t of $[n]$. Then, for all $j \in [\ell]$, $i \in [n]$, and $\bar{\tau} \in \{0,1\}^n$ it holds that

$$\begin{aligned}
\prod_{i \in \mathcal{J}} d_i^{\lambda_i^{\mathcal{J}}} &= \prod_{i \in \mathcal{J}, \tau_i = 0} d_i^{\lambda_i^{\mathcal{J}}} \cdot \prod_{i \in \mathcal{J}, \tau_i = 1} d_i^{\lambda_i^{\mathcal{J}}} \\
&= \prod_{i \in \mathcal{J}, \tau_i = 0} (Y_j)^{s_{j,i} \lambda_i^{\mathcal{J}} / y_j} \cdot \prod_{i \in \mathcal{J}, \tau_i = 1} \left(Z_j^r\right)^{s_{j,i} \lambda_i^{\mathcal{J}} / z_j} \\
&= \prod_{i \in \mathcal{J}, \tau_i = 0} (g^{y_j r})^{s_{j,i} \lambda_i^{\mathcal{J}} / y_j} \cdot \prod_{i \in \mathcal{J}, \tau_i = 1} (g^{z_j r})^{s_{j,i} \lambda_i^{\mathcal{J}} / z_j} \\
&= \prod_{i \in \mathcal{J}} (g^r)^{\lambda_i^{\mathcal{J}} s_{j,i}} = g^{r \cdot \sum_{i \in \mathcal{J}} \lambda_i^{\mathcal{J}} s_{j,i}} = g^{x_j r} = X_j^r = k.
\end{aligned}$$

And so every subset of t parties can decapsulate the correct key k, regardless of how many left keys and how many right keys there are.

One side security. The one side security of the scheme is based on the hardness of the Decisional Diffie-Hellaman (DDH) problem relative to $\mathcal{G}$.

Definition 7. *Let $\mathcal{G}$ be a group generation algorithm. We say that the Decisional Diffie-Hellaman (DDH) problem is hard relative to $\mathcal{G}$ if for every probablistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that*

$$\mathsf{Adv}_{\mathcal{A},\mathcal{G}}^{\mathrm{ddh}}(\lambda) := \left|\Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^{xy}) = 1\right] - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^z) = 1\right]\right| < \nu(\lambda)$$

for all $\lambda \in \mathbb{N}$, where $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ and $x, y, z \xleftarrow{\$} \mathbb{Z}_p$.

Theorem 3. *For every probabilistic polynomial-time adversary $\mathcal{A}_1$ there exists a probabilistic polynomial-time adversary $\mathcal{A}_2$ such that*

$$\mathsf{Adv}_{\mathcal{A}_2,\mathcal{G}}^{\mathrm{ddh}}(\lambda) = \mathsf{Adv}_{\mathcal{A}_1,\mathcal{BTDDH}}^{\mathrm{oss}}(\lambda)$$

for all $\lambda \in \mathbb{N}$.

Proof. The adversary $\mathcal{A}_2$ gets as input a tuple $(\mathbb{G}, p, g, X, Y, Z)$, where $X = g^x$ and $Y = g^y$ for uniformly random $x, y \in \mathbb{Z}_p$, and Z is either g^{xy} or g^z for a uniformly random $z \in \mathbb{Z}_q$. $\mathcal{A}_2$ first invokes $\mathcal{A}_1(1^\lambda)$ to obtain n, t, ℓ, u and d. We first discuss the case $d = 0$. In this case, $\mathcal{A}_2$ samples keys for $\mathcal{BTDDH}$ and a ciphertext-key pair as follows:

- For positions $j \neq u$ $\mathcal{A}_2$, samples all keys honestly. That is, it samples $x_j, y_j, z_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $X_j \leftarrow g^{x_j}$, $Y_j \leftarrow g^{y_j}$, and $Z_j \leftarrow g^{z_j}$. It then samples a Shamir t-out-of-n secret sharing $s_{j,1}, \ldots, s_{j,n}$ of x_j, and sets $pk_j \leftarrow (X_j, Y_j, Z_j)$, $sk_{i,0}^{(j)} \leftarrow s_{j,i}/y_j$ and $sk_{i,1}^{(j)} \leftarrow s_{j,i}/z_j$.
- For position u, $\mathcal{A}_2$ samples and computes $x_u, y_u \xleftarrow{\$} \mathbb{Z}_p$, and computes X_u, Y_u, and $sk_{1,0}^{(u)} = s_{u,1}/y_u, \ldots, sk_{n,0}^{(u)} = s_{u,n}/y_u$ as above. It also sets $Z_u \leftarrow X$ and $pk_u \leftarrow (X_u, Z_u)$. $\mathcal{A}_2$ sets $pk \leftarrow (pk_1, \ldots, pk_n)$ and $pkc \leftarrow \perp$.
- $\mathcal{A}_2$ sets $c_0 \leftarrow Y^{y_u}$ and $c_1 \leftarrow Z$ and then sets the ciphertext to be $c \leftarrow (c_0, c_1)$. It also sets $k^{(0)} \leftarrow Y^{x_u}$.

$\mathcal{A}_2$ then passes $pk, pkc, (c_0, c_1), k^{(0)}, \{(sk_{1,0}^{(j)}, sk_{1,1}^{(j)}), \ldots, (sk_{n,0}^{(j)}, sk_{n,1}^{(j)})\}_{j\in[\ell]\setminus\{u\}}$ and $(sk_{1,0}^{(u)}, \ldots, sk_{n,0}^{(u)})$ to $\mathcal{A}_1$, who replies with a bit b'. $\mathcal{A}_2$ then outputs the same bit b'. The reduction is symmetric in case $d = 1$.

Consider two cases. If $Z = g^{xy}$, then $\mathcal{A}_2$ simulates to $\mathcal{A}_1$ the experiment $\mathbf{ExpBTKEM}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda)$ with $b = 1$. On the other hand, if Z is a uniformly random group element, then $\mathcal{A}_2$ simulates to $\mathcal{A}_1$ the security experiment $\mathbf{ExpBTKEM}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda)$ with $b = 0$. It follows that

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{A}_2,\mathcal{G}}(\lambda) &= |\Pr[\mathcal{A}_2(\mathbb{G}, p, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{A}_2(\mathbb{G}, p, g, g^x, g^y, g^z) = 1]| \\
&= \left|\Pr\left[b' = 1 \;\middle|\; b = 1\right] - \Pr\left[b' = 1 \;\middle|\; b = 0\right]\right| \\
&= \left|\Pr\left[b' = 1 \;\middle|\; b = 1\right] - \left(1 - \Pr\left[b' = 0 \;\middle|\; b = 0\right]\right)\right| \\
&= \left|2\Pr\left[b' = b\right] - 1\right| \\
&= \left|2\Pr\left[\mathbf{ExpBTKEM}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda) = 1\right] - 1\right| \\
&= \mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda).
\end{aligned}$$

This concludes the proof.

Semantic security. We now argue the semantic security of the scheme, which is also based on the DDH assumption.

Theorem 4. *For every probablistic polynpmial-time adversary $\mathcal{A}_1$ there exists a probablistic polynpmial-time adversary $\mathcal{A}_2$ such that*

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{A}_2,\mathcal{G}}(\lambda) = \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda)$$

for all $\lambda \in \mathbb{N}$.

Proof. The adversary $\mathcal{A}_2$ gets as input a tuple $(\mathbb{G}, p, g, A, B, C)$, where $A = g^\alpha$ and $Y = g^\beta$ for uniformly random $\alpha, \beta \in \mathbb{Z}_p$, and Z is either $g^{\alpha\beta}$ or g^γ for a uniformly random $\gamma \in \mathbb{Z}_q$. $\mathcal{A}_2$ first invokes $\mathcal{A}_1(1^\lambda)$ to obtain n, t, ℓ.

$\mathcal{A}_2$ samples a position $u \xleftarrow{\$} [\ell]$ and continues in the simulation of the **IND-CPA** experiment as follows:

- $\mathcal{A}_2$ samples a public key for $\mathcal{BTBF}$ by sampling $x_1, \ldots, x_{u-1}, x_{u+1}, \ldots, x_\ell \xleftarrow{\$} \mathbb{Z}_p$ and $y_1, \ldots, y_\ell$, $z_1, \ldots, y_\ell \xleftarrow{\$} \mathbb{Z}_p$. It then sets $X_u \leftarrow A$ and $X_j \leftarrow g^{x_j}$ for every $j = 1, \ldots, u-1, u+1, \ldots, \ell$, and $Y_j \leftarrow g^{y_j}$, $Z_j \leftarrow g^z$ for every $j \in [\ell]$. It sets $pk_j \leftarrow (X_j, Y_j, Z_j)$ for every j and sends $pk \leftarrow (pk_1, \ldots, pk_\ell)$ and $pkc \leftarrow \perp$ to $\mathcal{A}_1$, who may then issue corruption queries.
- Assume without loss of generaility that $\mathcal{A}_1$ never repeats queries and always issues at most $t-1$ queries (otherwise, the output of the semantic security game is 0 no matter what). $\mathcal{A}_2$ samples $s_1, \ldots, s_{t-1} \xleftarrow{\$} \mathbb{Z}_p$ and for $q = 1, \ldots, Q$, it replies to the qth corruption queries of $\mathcal{A}_1$ as follows. Let $i \in [n]$ be the party corrupted in the qth query. The secret key of party i is composed of 2ℓ components, a left secret key and a right one in each of the ℓ positions:
  - For positions $j \neq u$, $\mathcal{A}_2$ computes the secret keys honestly using its knowledge of x_j, y_j and z_j.
  - For position u, $\mathcal{A}_2$ sets s $sk_{i,0}^{(u)} \leftarrow s_q/y_u$ and $sk_{i,1}^{(u)} \leftarrow s_q/z_u$.

  $\mathcal{A}_2$ then replies to $\mathcal{A}_1$ with $\{(sk_{i,0}^{(j)}, sk_{i,1}^{(j)})\}_{j\in[\ell]}$. Note that the secret keys are distributed as honestly-generated keys of $\mathcal{BTDDH}$. In particular, since $\mathcal{A}_1$ is guaranteed to query at most $Q \leq t-1$ queries, for any Q-tuple of parties $(i_1, \ldots, i_Q)$, the Shamir t-out-of-n secret shares of x_u are distributed uniformly in $\mathbb{F}_p^Q$.

– $\mathcal{A}_2$ computes the challenge ciphertext at position u by $c_0 \leftarrow B^y, c_1 \leftarrow B^z$ and sets the challenge key to be $k \leftarrow V$. It then passes $c = (c_0, c_1)$ and k as the challenge to $\mathcal{A}_1$. If $\mathcal{A}_1$ issues additional corruption queries after obtaining c and k, $\mathcal{A}_2$ replies to them as before.
– Finally, $\mathcal{A}_1$ outputs a bit b'. $\mathcal{A}_2$ outputs b' as well.

Observe that $\mathcal{A}_2$ simluates $\textbf{IND-CPA}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda)$ perfectly to $\mathcal{A}_1$. Moreover, if $V = g^{\alpha\beta}$ then $\mathcal{A}_2$ simulates the experiment $\textbf{IND-CPA}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda)$ with the bit $b = 1$; that is, it passes to $\mathcal{A}_1$ a key sampled from the correct distribution according to $\mathcal{BTDDH}$. On the other hand, if V is a uniformly random element of $\mathbb{G}$, $\mathcal{A}_2$ simulates $\textbf{IND-CPA}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda)$ with the bit $b = 0$. Hence, it holds that

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{A}_2,\mathcal{G}}(\lambda) = \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A}_1,\mathcal{BTDDH}}(\lambda).$$

This concludes the proof.

5.2 Constant-Size Ciphertexts and Public Keys from Pairings

In this section, we show that by relying on pairings, one can obtain a BT-KEM scheme with a constant-size public key, in addition to a constant-size ciphertext.

Formally, we consider a bilinear group generation algorithm $\mathcal{G}$ that takes in the security parameter and outputs a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, p)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are descriptions of cyclic groups of order p, where $\mathbb{G}_b$ is generated by g_b for $b = 0, 1$. The function e is a non-degenerate bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$. The construction also relies on the existence of a hash function $H : \mathbb{N} \to \mathbb{G}_2$. Formally, the function H should depend on $\mathbb{G}_2$ but we ignore this in our notation for convenience and fix a single function H for all groups. The scheme is formally defined in Fig. 8.

Correctness. Let $0 < t < n$ and let $\ell > 0$. Let $\mathcal{J}$ be a subset of size t of $[n]$, and for all $j \in [\ell]$, let w_j denote the $\mathbb{Z}_p$ element for which $H(j) = g_2^{w_j}$. Then, for all $j \in [\ell]$, $i \in [n]$, and $\bar{\tau} \in \{0,1\}^n$ it holds that

$$\begin{aligned}
\prod_{i\in\mathcal{J}} d_i^{\lambda_i^{\mathcal{J}}} &= \prod_{i\in\mathcal{J},\tau_i=0} d_i^{\lambda_i^{\mathcal{J}}} \cdot \prod_{i\in\mathcal{J},\tau_i=1} d_i^{\lambda_i^{\mathcal{J}}} \\
&= \prod_{i\in\mathcal{J},\tau_i=0} e(Y^r, H(j)^{s_i z})^{\lambda_i^{\mathcal{J}}} \cdot \prod_{i\in\mathcal{J},\tau_i=1} e(Z^r, H(j)^{s_i y})^{\lambda_i^{\mathcal{J}}} \\
&= \prod_{i\in\mathcal{J},\tau_i=0} e(g_1^{yr}, g_2^{w_j s_i z})^{\lambda_i^{\mathcal{J}}} \cdot \prod_{i\in\mathcal{J},\tau_i=1} e(g_1^{zr}, g_2^{w_j s_i y})^{\lambda_i^{\mathcal{J}}} \\
&= \prod_{i\in\mathcal{J}} e(g_1, g_2)^{w_j r y z \lambda_i^{\mathcal{J}} s_i} = e(g_1, g_2)^{w_j r y z \cdot \sum_{i\in\mathcal{J}} \lambda_i^{\mathcal{J}} s_i} = e(g_1, g_2)^{w_j r x y z} \\
&= e(g_1^{xyzr}, g_2^{w_j}) = e(X^r, H(j)) = k.
\end{aligned}$$

And so every subset of t parties can decapsulate the correct key k, regardless of how many left keys and how many right keys there are.

One side security. We now prove the one side security of $\mathcal{BTBF}$. One side security follows from a variant of the DDH assumption in the source group $\mathbb{G}_1$. Note that by making this assumption, we are assuming in particular that there is no efficiently-computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$.

A BT-KEM scheme $\mathcal{BTBF}$

Public parameters: $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, p)$ sampled by $\mathcal{G}(1^\lambda)$.

$\mathcal{BTBF}$.**KeyGen**$(1^\lambda, n, t, \ell)$**:**

1. Sample $x, y, z \xleftarrow{\$} \mathbb{Z}_p$ and compute $X \leftarrow g_1^{xyz}$, $Y \leftarrow g_1^y$, and $Z \leftarrow g_1^z$.
2. Sample a Shamir t-out-of-n secret sharing $s_1, \ldots, s_n$ of x.
3. For $j = 1, \ldots, \ell$ do:
   (a) Compute $h_j \leftarrow H(j) \in \mathbb{G}_2$.
   (b) Set $sk_{i,0}^{(j)} \leftarrow h_j^{s_i z}$ and $sk_{i,1}^{(j)} \leftarrow h_j^{s_i y}$ for $i = 1, \ldots, n$.
4. Set $pk \leftarrow (X, Y, Z)$ and $pkc \leftarrow \perp$.
5. Return $(pk, pkc, \{(sk_{1,0}^{(j)}, sk_{1,1}^{(j)}), \ldots, (sk_{n,0}^{(j)}, sk_{n,1}^{(j)})\}_{j \in [\ell]})$.

$\mathcal{BTBF}$.**Enc**(pk, j)**:**

1. Parse pk as (X, Y, Z).
2. Compute $h_j \leftarrow H(j) \in \mathbb{G}_2$.
3. Sample $r \xleftarrow{\$} \mathbb{Z}_p$ and set $k \leftarrow e(X^r, h_j)$.
4. Compute $c_0 \leftarrow Y^r$ and $c_1 \leftarrow Z^r$ and set $c \leftarrow (c_0, c_1)$
5. Return (k, c).

$\mathcal{BTBF}$.**Dec**$(j, sk_{i,b}^{(j)}, c = (c_0, c_1))$**:**

1. Compute $d_i \leftarrow e(c_b, sk_{i,b}^{(j)})$.
2. Return d_i.

$\mathcal{BTBF}$.**Combine**$(pkc, j, c, \mathcal{J}, \{d_i\}_{i \in \mathcal{J}})$**:**

1. Compute the Lagrange coefficients $(\lambda_i^{\mathcal{J}})_{i \in \mathcal{J}}$ corresponding to the subset $\mathcal{J}$: $\lambda_i^{\mathcal{J}} \leftarrow \prod_{v \in \mathcal{J} \setminus \{i\}} \frac{v}{v-i} \in \mathbb{Z}_p$.
2. Compute $k \leftarrow \prod_{i \in \mathcal{J}} d_i^{\lambda_i^{\mathcal{J}}}$.
3. Return k.

Fig. 8. Our BT-KEM scheme in pairing groups, denoted $\mathcal{BTBF}$.

Definition 8. *Let $\mathcal{G}$ be a bilinear group generation algorithm. We say that the Augmented Decisional Diffie-Hellman problem (ADDH) is hard relative to the source group of $\mathcal{G}$ if for every probabilistic polynomial-time algorithm $\mathcal{A}$ the following function*

$$\mathsf{Adv}^{\mathrm{addh1}}_{\mathcal{A},\mathcal{G}}(\lambda) := \left| \begin{array}{l} \Pr\left[\mathcal{A}(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,p,\vec{h},g_1^{yz}) = 1\right] \\ \qquad - \Pr\left[\mathcal{A}(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,p,\vec{h},g_1^{v}) = 1\right] \end{array} \right|$$

is negligible in $\lambda \in \mathbb{N}$, where $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p,g_1,g_2,e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ and $\vec{h} = (g_1, g_2, g_1^x, g_1^y, g_1^z, g_1^{xy}, g_1^{xz}, e(g_1,g_2)^{xyz})$ for $x,y,z,v \xleftarrow{\$} \mathbb{Z}_p$.

The ADDH problem defined above can be seen as a special case of the Uber problem in bilinear groups [8,22]. Also note that the "target element" g_1^{yz} cannot be trivially computed via group operations and the pairing operation from the other input elements, namely the elements in $\vec{h}$. Therefore, the ADDH problem is hard in the generic group model [51,8] and its hardness is reducible to that of the standard discrete log problem in the (decisional) algebraic group model [31,49,3,48].

The following theorem reduces the one side security of $\mathcal{BTBF}$ to the hardness of ADDH in $\mathbb{G}_1$.

Theorem 5. *For every probabilistic polynomial-time adversary $\mathcal{A}_1$ there exists a probabilistic polynomial-time adversary $\mathcal{A}_2$ such that*

$$\mathsf{Adv}^{\mathrm{addh1}}_{\mathcal{A}_2,\mathcal{G}}(\lambda) = \mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda)$$

for all $\lambda \in \mathbb{N}$, when the hash function H is modeled as a random oracle.

Proof. The adversary $\mathcal{A}_2$ gets $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,p,g_1,g_2,A,B,C,P1,P2,P3,V)$ as input, where $A = g_1^\alpha$, $B = g_1^\beta$, $C = g_1^\gamma$, $P1 = g_1^{\alpha\beta}$, $P2 = g_1^{\alpha\gamma}$ and $P3 = e(g_1,g_2)^{\alpha\beta\gamma}$ for uniformly random $\alpha,\beta,\gamma \in \mathbb{Z}_p$, and V is either $g_1^{\beta\gamma}$ or a uniformly random element of $\mathbb{G}_1$.

Assume without loss of generality that $\mathcal{A}_1$ never repeats queries to the random oracle. Then, $\mathcal{A}_2$ first invokes $\mathcal{A}_1(1^\lambda)$ to obtain n, t, ℓ, u and d.

To simulate the random oracle H to $\mathcal{A}_1$, $\mathcal{A}_2$ does:

- For any query $j \in \mathbb{N}$, $\mathcal{A}_2$ samples a uniformly random $w_j \xleftarrow{\$} \mathbb{Z}_p$ and replies with $h_j \leftarrow g_2^{w_j}$.
- Once $\mathcal{A}_1$ has outputted the integers u and ℓ, for all indices $j \in [\ell]$ for which $H(j)$ has not been previously queried for by $\mathcal{A}_1$, $\mathcal{A}_2$ samples a uniformly random $w_j \xleftarrow{\$} \mathbb{Z}_p$ and sets with $h_j \leftarrow g_2^{w_j}$. If $\mathcal{A}_1$ subsequently queries for $H(j)$, then $\mathcal{A}_2$ replies with h_j.

As before, we describe the remainder of the reduction for the case $d = 0$ and the case $d = 1$ is defined analogously. Upon receiving n, t, ℓ, u and d, $\mathcal{A}_2$ samples keys for $\mathcal{BTBF}$ and a key-ciphertext pair as follows:

- $\mathcal{A}_2$ samples $x \xleftarrow{\$} \mathbb{Z}_p$ and sets $X \leftarrow P_1^x$, $Y \leftarrow A$, $Z \leftarrow B$, and $pk \leftarrow (X,Y,Z)$. It also samples t-out-of-n Shamir secret shares of x, denote them by $s_1,\ldots,s_n$.
- Left secret keys are generated honestly using knowledge of $w_1,\ldots,w_\ell$, the secret shares $s_1,\ldots,s_n$, and access to the element B. That is, $\mathcal{A}_2$ sets $sk_{i,0}^{(j)} \leftarrow B^{s_i w_j}$. Note that these keys are indeed distributed correctly since $B^{s_i w_j} = h_j^{s_i \beta}$ and $Z = B$. Right secret keys are generated similarly, by computing $sk_{i,1}^{(j)} \leftarrow A^{s_i w_j}$.
- $\mathcal{A}_2$ sets $c_0 \leftarrow P2$, $c_1 \leftarrow V$, and $c \leftarrow (c_0, c_1)$. It also sets $k \leftarrow P3^{x w_u}$.

$\mathcal{A}_2$ then passes $pk, pkc, (c_0, c_1), k, \{(sk_{1,0}^{(j)}, sk_{1,1}^{(j)}), \ldots, (sk_{n,0}^{(j)}, sk_{n,1}^{(j)})\}_{j\in[\ell]\setminus\{u\}}$ and $(sk_{1,0}^{(u)}, \ldots, sk_{n,0}^{(u)})$ to $\mathcal{A}_1$, who replies with a bit b'. $\mathcal{A}_2$ then outputs the same bit b'. The reduction is symmetric in case $d = 1$. Note that when $C = g_1^{\beta\gamma}$, then the ciphertext is of the form $(g_1^{\alpha\gamma}, g_1^{\beta\gamma}) = (Y^\gamma, Z^\gamma)$ for a uniformly random $\gamma \in \mathbb{Z}_p$. Hence, it is distributed like a valid ciphertext (the case $b = 1$ in the security experiment $\mathbf{ExpBTKEM}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda)$). Otherwise, c_1 is a uniformly random group element, independent of c_0 (the case $b = 0$ in the $\mathbf{ExpBTKEM}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda)$ security experiment). Hence, by the same analysis as in the proof of Theorem 3, it holds that

$$\mathsf{Adv}^{\mathrm{ddh1}}_{\mathcal{A}_2,\mathcal{G}}(\lambda) = \mathsf{Adv}^{\mathrm{oss}}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda),$$

concluding the proof of the theorem.

Semantic security. We now argue the semantic security of $\mathcal{BTBF}$, relying on the standard Decisional Bilinear Diffie-Hellaman (DBDH) assumption.

Definition 9. *Let $\mathcal{G}$ be a bilinear group generation algorithm. We say that the Decisional Bilinear Diffie-Hellaman (DBDH) problem is hard relative to $\mathcal{G}$ if for every probabilistic polynomial-time algorithm $\mathcal{A}$ the following function*

$$\mathsf{Adv}^{\mathrm{dbdh}}_{\mathcal{A},\mathcal{G}}(\lambda) := \left| \begin{array}{l} \Pr\left[\mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e, g_1^x, g_1^y, g_2^z, e(g_1, g_2)^{xyz}) = 1\right] \\ \qquad - \Pr\left[\mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e, g_1^x, g_1^y, g_2^z, e(g_1, g_2)^{v}) = 1\right] \end{array} \right|$$

is negligible in $\lambda \in \mathbb{N}$, where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ and $x, y, z, v \xleftarrow{\$} \mathbb{Z}_p$.

The following theorem establishes the semantic security of $\mathcal{BTBF}$.

Theorem 6. *For every probabilistic polynomial-time adversary $\mathcal{A}_1$ there exists a probabilistic polynomial-time adversary $\mathcal{A}_2$ such that*

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda) \leq Q \cdot \mathsf{Adv}^{\mathrm{dbdh}}_{\mathcal{A}_2,\mathcal{G}}(\lambda)$$

for all $\lambda \in \mathbb{N}$, when the hash function H is modeled as a random oracle and $Q = Q(\lambda)$ is an upper bound on the number of oracle queries issued by $\mathcal{A}_1$.

Proof. The adversary $\mathcal{A}_2$ gets as input a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e, A, B, C, V)$, where $A = g_1^\alpha$, $B = g_1^\beta$, $C = g_2^\gamma$ for uniformly ranodm $\alpha, \beta, \gamma \in \mathbb{Z}_p$, and V is either $e(g_1, g_2)^{\alpha\beta\gamma}$ or $e(g, g)^v$ for a uniformly random $v \in \mathbb{Z}_q$. Let Q denote an upper bound on the number of oracle queries issued by $\mathcal{A}_1$ and assume without loss of generality that $\mathcal{A}_1$ never repeats queries. Then, $\mathcal{A}_2$ first invokes $\mathcal{A}_1(1^\lambda)$ to obtain n, t and ℓ.

To simulate the random oracle H to $\mathcal{A}_1$, $\mathcal{A}_2$ samples an index $q^* \xleftarrow{\$} [Q]$ and does:

- On the q^*th query of $\mathcal{A}_1$: let $j^* \in \mathbb{N}$ be the query. Then, $\mathcal{A}_2$ sets $h_{j^*} \leftarrow C$ and replies with its input element h_{j^*}.
- For any other query $j \in \mathbb{N}$, $\mathcal{A}_2$ samples a uniformly random $w_j \xleftarrow{\$} \mathbb{Z}_p$ and replies with $h_j \leftarrow g^{w_j}$.
- Once $\mathcal{A}_1$ has outputted the integers u and ℓ, if h_u is already set to an element other than C, then $\mathcal{A}_2$ aborts. Denote this event by abort. If not aborted, then for all indices $j \in [\ell]$ for which $H(j)$ was not previously queries for by $\mathcal{A}_1$, $\mathcal{A}_2$ samples a uniformly random $w_j \xleftarrow{\$} \mathbb{Z}_p$ and sets with $h_j \leftarrow g^{w_j}$. If $\mathcal{A}_1$ subsequently queries for $H(j)$, then $\mathcal{A}_2$ replies with h_j.

Upon receiving n, t, ℓ, $\mathcal{A}_2$ continues in the simulation of the **IND-CPA** experiment as follows:

- samples a public key for $\mathcal{BTBF}$ by sampling $y, z \xleftarrow{\$} \mathbb{Z}_p$ and setting $X \leftarrow A^{yz}$, $Y \leftarrow g_1^y$, $Z \leftarrow g_1^z$ and $pk \leftarrow (X, Y, Z)$. It sends pk and $pkc \leftarrow \perp$ to $\mathcal{A}_1$, who may then issue corruption queries.
- Assume without loss of generaility that $\mathcal{A}_1$ never repeats queries and always issues at most $t-1$ queries (otherwise, the output of the semantic security game is 0 no matter what). $\mathcal{A}_2$ samples $s_1, \ldots, s_{t-1} \xleftarrow{\$} \mathbb{Z}_p$ and for $q = 1, \ldots, Q$, it replies to the qth corruption queries of $\mathcal{A}_1$ as follows. Let $i \in [n]$ be the party corrupted in the qth query. The secret key of party i is composed of 2ℓ components, a left secret key and a right one in each of the ℓ positions. $\mathcal{A}_2$ computes these honestly from $h_1, \ldots, h_\ell$, using its knowledge of y and z, and replacing party i's secret share of x with the uniformly sampled s_q. That is, for $j = 1, \ldots, \ell$: $\mathcal{A}_2$ sets $sk_{i,0}^{(j)} \leftarrow h_j^{s_q z}$ and $sk_{i,1}^{(j)} \leftarrow h_j^{s_q y}$. $\mathcal{A}_2$ then replies to qth corruption query with $\{(sk_{i,0}^{(j)}, sk_{i,1}^{(j)})\}_{j \in [\ell]}$.
  Note that the secret keys are distributed as honestly generated keys of $\mathcal{BTBF}$. In particular, since $\mathcal{A}_1$ is guaranteed to query at most $Q \leq t-1$ queries, for any Q-tuple of parties $(i_1, \ldots, i_Q)$, the Shamir t-out-of-n secret shares of x are distributed uniformly in $\mathbb{F}_p^Q$.
- To compute the challenge to $\mathcal{A}_1$, $\mathcal{A}_2$ computes the challenge ciphertext $c_0 \leftarrow B^y, c_1 \leftarrow B^z$ and sets the challenge key to be $k \leftarrow V^{yz} \in \mathbb{G}_T$. It then passes $c = (c_0, c_1)$ and k as the challenge to $\mathcal{A}_1$. If $\mathcal{A}_1$ issues additional corruption queries after obtaining c and k, $\mathcal{A}_2$ replies to them as before.
- Finally, $\mathcal{A}_1$ outputs a bit b'. $\mathcal{A}_2$ outputs b' as well.

Observe that $\mathcal{A}_2$ simluates $\textbf{IND-CPA}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda)$ perfectly to $\mathcal{A}_1$ as long as the event abort defined above does not occur. Moreover, if $V = e(g_1, g_2)^{\alpha\beta\gamma}$ then $\mathcal{A}_2$ simulates the experiment $\textbf{IND-CPA}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda)$ with the bit $b = 1$; that is, it passes to $\mathcal{A}_1$ a key sampled from the correct distribution according to $\mathcal{BTBF}$. On the other hand, if V is a uniformly random element of $\mathbb{G}_T$, $\mathcal{A}_2$ simulates $\textbf{IND-CPA}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda)$ with the bit $b = 0$. Hence, it holds that

$$\begin{aligned} \mathsf{Adv}^{\mathrm{dbdh}}_{\mathcal{A}_2,\mathcal{G}}(\lambda) &\geq \Pr\left[\mathsf{abort}\right] \cdot \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda) \\ &\geq \frac{1}{Q}\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{A}_1,\mathcal{BTBF}}(\lambda). \end{aligned}$$

This concludes the proof.

6 The Case of Small Traitor Coalitions

In this section, we consider the setting in which the traitor coalition is small; namely, there are $f < t$ traitors, where t is the decryption threshold as before. We begin with a brief overview of the challenges that this setting poses and ways to circumvent them. In this informal discussion, we omit many technical details that can be found later in this section.

In the small coalition setting, we would like to capture the scenario in which the traitor coalition publishes or sells any piece of information that might be useful for decryption by another set of parties. To be as general as possible, we model this information as a decoder D that receives additional decryption shares for some ciphertext c. When the decoder receives enough information to decrypt, it outputs the encrypted message m. The decoder may be tied to a specific ciphertext c and only decrypt this ciphertext, or it may be a general decoder that gets also a ciphertext c as input and tries to decrypt it.[3] We focus on decoders that are tied to a specific ciphertext, but the same problems and solutions arise in both cases. As a concrete example, think of a decoder D that has a ciphertext c and the secret keys of the $f < t$ corrupted parties embedded in it. Whenever it gets $t - f$ decryption shares from *additional* parties as input, it decrypts and outputs the message. If it does not get enough information to decrypt, it outputs $\perp$. Given black-box access to such a decoder, we would like to trace it back to at least one of the corrupted parties.

Unfortunately, if we allow for arbitrary decoders, then for robust schemes (where in particular, shares can be publicly traced to their generating party), efficient tracing becomes hopeless for most choices of f and t. The issue arises already with the simple decoder D described above that has a subset $\mathcal{I}$ of f keys embedded in it, if it is restricted to accept *exactly* $t - f$ shares as input. If it gets more shares than that as input, it refuses to decrypt and outputs $\perp$. Denote this decoder by $D_{\mathcal{I}}$. Intuitively, to extract *any meaningful information* from the decoder, we need to make it work; i.e., output something other than $\perp$. Now consider the distribution over decoders that are defined as above, where the subset $\mathcal{I}$ is sampled uniformly at random from all subsets of $[n]$ of size f. In this case, we prove the following lower bound.

Theorem 7 (informal). *If the underlying threshold decryption scheme is robust, then for any algorithm $\mathcal{B}$ making at most Q oracle queries to $D_{\mathcal{I}}$, the probability that $D_{\mathcal{I}}$ returns anything but $\perp$ is bounded by* $\frac{Q}{\binom{n}{f} - Q \cdot \binom{n-(t-f)}{f}}$.

As an example, consider the setting where $t = n/3$ and $f = t/2$. In this case, the probability of making $D_{\mathcal{I}}$ output something other than $\perp$ is bounded by $Q \cdot 2^{-\Omega(n)}$. Hence, an exponential number of queries to $D_{\mathcal{I}}$ are required to even make it work. In Appendix B, we make this theorem precise and provide a proof.

In light of the above, we explore two restricted classes of decoders that still capture meaningful traitor attacks on the one hand, yet allow for some meaningful notion of accountability on the other hand.

Option I: A decoder that can decrypt, must. The reason the decoder D sketched above was untraceable, was that it could choose to reject any input that consisted of more than $t - f$ shares. A natural question is then what happens if we disallow such behavior on the side of the decoder? Concretely, we consider here a restricted class of decoders: a decoder $D = D(\mathcal{I})$ is associated with some subset $\mathcal{I}$ of corrupted parties, and if the decoder is fed decryption shares from a subset $\mathcal{S}$ of parties, it outputs the correct message whenever $|\mathcal{I} \cup \mathcal{S}| \geq t$. This means that the decoder may output the correct message also on inputs that include up to t, and in particular more than $t - f$, decryption shares. We call such decoders *universal*. Though (unavoidably) restricted, this class of decoders already captures natural forms of information leakage, such as leaking the secret keys of the parties in $\mathcal{I}$ or their decryption shares of a specific ciphertext c (or malformed keys/decryption shares, that still allow decryption).

We show that with this restriction on the decoder, tracing becomes possible. We present a tracing procedure that can be seen as the mirror image of tracing using private linear broadcast encryption [16]. By semantic security, we know that when the decoder D gets no decryption share as input, it should output the correct message with negligible probability. On the other hand, when

[3] Note that in the case where $f \geq t$ it makes no sense to consider decoders that are tied to a specific ciphertext c. This is because when $f \geq t$ the traitor coalition can just decrypt c and publish the underlying message m as the decoder. This "decoder" is trivially untraceable.

given any t decryption shares $d_1, \ldots, d_t$ as input, any universal decoder should output the correct message, say with probability 1 (below, we relax this notion and consider probabilistic decoders). This means that there must be some index $i \in [t]$ such that $D(d_1, \ldots, d_i)$ outputs the correct message, but $D(d_1, \ldots, d_{i-1})$ does not. Since D is universal, we may deduce that $\{d_1, \ldots, d_{i-1}\}$ is insufficient information for D to decrypt, but $\{d_1, \ldots, d_i\}$ is. Hence it must be the case that party i is innocent, in the sense that it is not a member of the subset $\mathcal{I}$ of corrupted parties that define the universal decoder $D = D(\mathcal{I})$. Feeding the subsets $\emptyset, \{d_1\}, \{d_1, d_2\}, \ldots, \{d_1, \ldots, d_t\}$ to D one by one, we are guaranteed to find this i. Moreover, we show that by repeating this procedure, we can "exonerate" all innocent parties in $\{1, \ldots, t\}$. The full tracing algorithm is obtained by doing the same for all subsets $\{1, \ldots, t\}$, $\{t+1, \ldots, 2t\}, \ldots, \{n-t+1, \ldots, n\}$.

Option II: Confirmation instead of tracing. The tracing procedure above crucially relied on the fact that the decoder works for inputs consisting of a variable number of decryption shares. More clever decoders might try to avoid detection by insisting that the decoder takes in *exactly* $t - f$ decryption shares as input. This can be done, for example, by hard-coding the traitors' secret keys to the decryption circuit and obfuscating it. By the lower bound result sketched above, in this setting we cannot hope for full-fledged traitor tracing. Instead, we explore the task of *"traitor confirmation"*: given knowledge of the traitor coalition $\mathcal{I}$ that manufactured some decoder D (and possibly some trapdoor information) the task is to create a publicly-verifiable proof that ascertains that this is indeed the set of traitors. Both the prover and the verifier are given black-box access to D. Importantly, a malicious prover should not be able to frame an innocent party for the creation of a decoder D. This should be the case even if the malicious prover gets access to more than t of the parties' secret keys. This is important, for example, to fend against cases in which a coalition creates some decoder D, and at a later stage obtains additional secret keys that it did not possess when manufacturing D. If now this coalition has more than t keys, they could potentially reconstruct the keys of all n parties. Framing an innocent party that did not partake in the creation of D should still be impossible.

Here, too, we consider decoders $D = D(\mathcal{I})$ that are specified by a subset $\mathcal{I}$ of corrupted parties, and must decrypt if they get decryption shares from a disjoint subset of parties $\mathcal{S}$ of size $t - f$ where $f = |\mathcal{I}|$.[4] But now, we do allow the decoder to take in exactly $t - f$ decryption shares as inputs, and refuse to work on more shares than that. We call such decoders *exact decoders*. This is exactly the decoder from the lower bound sketched above. Focusing on confirmation rather than tracing allows us to circumvent the lower bound.

Now say that we have a robust threshold decryption scheme, and want to produce a proof that a subset $\mathcal{I}$ of size $f < t$ is the corrupted subset behind some decoder D for a ciphertext c. The first idea that comes to mind is to include in the proof the decryption shares $\{d_j\}_{j \in [n] \setminus \mathcal{I}}$ of c of all parties outside of $\mathcal{I}$. To verify, one partitions these into $k \approx (n-f)/(t-f)$ subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k$, each of size $t - f$, feeds these to D one by one, and asserts that D outputs the correct message m every time. If $\mathcal{I}$ is indeed the corrupted subset underlying D, then verification will indeed go through. However, there is a problem. This confirmation system is completely insecure and is susceptible to framing! As a small example, consider the case $n = 6$ and $t = 4$, and an adversary that corrupts the subset $\mathcal{I} = \{1, 3, 5\}$ and constructs the decoder $D = D(\{1, 3, 5\})$. This adversary can later claim that the corrupted set of parties is in fact the set $\mathcal{J} = \{1, 4\}$ and as proof gives the decryption shares of the other parties $\pi = \{d_2, d_3, d_5, d_6\}$. Verification will now go through since both $D(d_2, d_3)$ and

[4] This restriction is necessary due to a lower bound similar to the one described above.

$D(d_5, d_6)$ have enough information to decrypt. The issue here is that the adversary can lie about the number f of corrupted parties underlying the decoder.

To fix this issue, the verification procedure needs to run a few more checks. For every subset $\mathcal{S}_j$, for every party $i \in \mathcal{S}_j$, and for every party i' in the claimed set of traitors $\mathcal{I}$, verification will run $D((\mathcal{S}_j \setminus \{i\}) \cup \{i'\})$. If this query returns the correct message m, then we will reject the proof as invalid. The intuition behind this check is that if the corrupted subset is actually larger than what is claimed by the adversarially conjured proof, then there is a decryption share by a corrupted party i^* in some $\mathcal{S}_j$. Hence, removing it from $\mathcal{S}_j$ (and replacing it with an element in $\mathcal{I}$ to keep the number of input shares the same) should not affect D's ability to decrypt.

6.1 Traitor Confirmation

We now define what it means for a threshold decryption scheme to allow confirmation of small traitor sets (less than the threshold t in size). Later, we will show how any robust threshold decryption scheme can be augmented with a traitor confirmation mechanism to defend against a restricted class of decoders.

As we discussed before, confirmation for arbitrary decoders (e.g., a decoder that only works on a few subsets of inputs) is impossible. Therefore, we focus on a class of decoders, that we call admissible decoders, defined below. The definition assumes that D is deterministic and perfect, and always outputs the same on a given input $d_{i_1}, d_{i_2}, \ldots$. After presenting our confirmation definition and scheme, we discuss how to generalize this definition and the construction to a certain class of probabilistic imperfect decoders.

Definition 10. *We say that a decoder $D_{\mathcal{I}}$ for a cipher text $c \xleftarrow{\$} \mathcal{E}.\mathsf{Enc}(pk, m)$ is* admissible with respect to a set *of parties $\mathcal{I} \subseteq [n]$, if the decoder decrypts successfully if and only if, it is given at least $t - |\mathcal{I}|$ decryption shares for parties in $[n] \setminus \mathcal{I}$. More formally,*

$$\forall \mathcal{S} \subseteq [n], |\mathcal{S} \cup \mathcal{I}| \geq t \iff D_{\mathcal{I}}(\{d_j\}_{j \in \mathcal{S}}) = m$$

where $d_i \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i, c)$ are decryption shares for c for any $i \in [n]$. We say that a decoder D is admissible *if there exists a set $\mathcal{I}$ such that the behavior of D is identical to that of $D_{\mathcal{I}}$. More formally, D is admissible with respect to $\mathcal{I}$ if,*

$$\forall S \subseteq [n], D(\{d_j\}_{j \in \mathcal{S}}) = m \iff D_{\mathcal{I}}(\{d_j\}_{j \in \mathcal{S}}) = m$$

This set $\mathcal{I}$ is called the effective traitor set *of the decoder D.*

Remark 1. When the underlying encryption scheme $\mathcal{E}$ is robust, we give the robustness proofs as an additional input to the decoder along with each decryption share.

We consider two types of admissible decoders, namely universal and exact. A *universal decoder* $D_{\mathcal{I}}$ admissible with respect to $\mathcal{I} \subseteq [n]$ decrypts correctly if given decryption shares of any set $\mathcal{S}$ such that $|\mathcal{S} \cup \mathcal{I}| \geq t$, where the size of $\mathcal{S}$ is allowed to be any number in $[t]$, i.e. $1 \leq |\mathcal{S}| \leq t$. On the other hand, an *exact decoder* admissible with respect to $\mathcal{I} \subseteq [n]$ and $k \in [t]$, outputs the correct decryption only when given decryption shares of a set $\mathcal{S}$ such that $|\mathcal{S} \cup \mathcal{I}| \geq t$ and $|\mathcal{S}| = k$. Note that, for both universal and exact decoders, for any set $\mathcal{S}$ for which $|\mathcal{S} \cup \mathcal{I}| < t$, the decoder will not decrypt correctly. We now move on to defining traitor confirmation for threshold decryption schemes with respect to exact admissible decoders.

A threshold decryption scheme with **traitor confirmation** (TDTC for short) is a threshold decryption scheme with two additional algorithms, Prove and Verify.

– Prove takes in the public key pk, an alleged subset of traitors $\mathcal{I} \subseteq [n]$, a target ciphertext c^*, and a confirmation key ck that is an additional output of KeyGen. It also gets oracle access to a decoder D that takes decryption shares as input. The output of Prove is a proof π.
– Verify takes as input the public key pk, the combiner public key pkc, the confirmation verification key vk' which is an additional output of KeyGen, the set $\mathcal{I}$, the ciphertext c^*, and a proof π. It, too, has oracle access to the decoder D. Verify outputs either 1 implying acceptance of the proof, or 0, implying rejection.

A secure TDTC satisfies two properties: **confirmation correctness** and **confirmation integrity**. Informally, confirmation correctness means that if Prove is given as input the subset $\mathcal{I}$ that actually manufactured the decoder D (in the sense that any $t-|\mathcal{I}|$ decryption shares of parties not in $\mathcal{I}$ will make D output the correct message m), then Prove should output a proof that will be accepted by Verify. This is captured by the security experiment $\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda)$ in Fig. 9. The experiment is parameterized by an efficiently-samplable distribution S over the message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ of $\mathcal{E}$. We also define the collision probability of S, $\mathsf{cp}(S)$, as the probability that two messages sampled from the distribution S will be equal. More formally,

$$\mathsf{cp}(S) = \Sigma_{m\in\mathcal{M}} \Pr[\hat{m} = m : \hat{m} \leftarrow\!\!\$\, S]^2$$

The notion of $\mathsf{cp}(S)$ is not used by the security definitions, but will become handy when analyzing the security of our scheme.

Definition 11. *We say that $\mathcal{E}$ has confirmation correctness if for every probabilistic polynomial time adversary $\mathcal{A}$, the following function is negligible in λ:*

$$\mathsf{Adv}^{\mathrm{cc}}_{\mathcal{A},\mathcal{E},S}(\lambda) := \Pr[\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1].$$

Confirmation integrity states that an adversary cannot generate a proof that incriminates party i, if this party did not participate in the construction of D. This is captured by the security experiment $\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda)$ in Fig. 10.

Definition 12. *We say that $\mathcal{E}$ has confirmation integrity if for every probabilistic polynomial time adversary $\mathcal{A}$, the following function is negligible in λ:*

$$\mathsf{Adv}^{\mathrm{ci}}_{\mathcal{A},\mathcal{E},S}(\lambda) := \Pr[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1].$$

6.2 A Traitor Confirmation System

We now build a generic traitor confirmation system $\mathcal{TDTC}$ for any robust threshold decryption scheme $\mathcal{E}$ and for exact decoders D admissible with respect to a set $\mathcal{I} \subseteq [n]$ and $k \in [t]$. More concretely, we show how to add confirmation algorithms to any robust threshold decryption scheme $\mathcal{E}$. The algorithms ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine}, \mathsf{ShareVf}$) remain unchanged. The confirmation key is simply the list of secret keys of all n parties, i.e. $ck \leftarrow \{sk_i\}_{i\in[n]}$, and the confirmation verification key is simply the verification key used for verifying decryption shares, i.e. $vk' \leftarrow vk$. The new confirmation algorithms, Prove and Verify, follow the outline we sketch at the beginning of this section, and are formally defined in Fig. 11.

We now prove confirmation correctness and integrity. Theorems 8 and 9 below reduce the correctness and integrity of the scheme to the semantic security and robustness of the underlying encryption scheme $\mathcal{E}$.

Experiment $\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda)$

$j \leftarrow 1$
$(n, t, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
$(pk, pkc, sk_1, \ldots, sk_n, ck, vk') \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t)$
$\mathcal{I} \xleftarrow{\$} \mathcal{A}(\mathsf{state}, pk, pkc)$
$(D, j^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{ct}()}(\mathsf{state}, \{sk_i\}_{i \in \mathcal{I}})$
if $j^* \notin [j-1] \ \vee \ |\mathcal{I}| \geq t \ \vee \ \mathcal{I} = \emptyset$ **then return** 0
if D is not exact admissible w.r.t. $(\mathcal{I}, t - |\mathcal{I}|)$ **then return** 0
$\pi \xleftarrow{\$} \mathsf{Prove}^D(pk, \mathcal{I}, c_{j^*}, ck)$
if $\mathsf{Verify}^D(pk, pkc, vk', \mathcal{I}, c_{j^*}, \pi) = 0$ **then return** 1 **else return** 0

Oracle $\mathsf{ct}()$

$m_j \xleftarrow{\$} S$
$c_j \xleftarrow{\$} \mathsf{Enc}(pk, m_j)$
$j \leftarrow j + 1$
return c_j

Fig. 9. The adversarial confirmation correctness experiment for a TDTC scheme $\mathcal{E}$ and an adversary $\mathcal{A}$. The experiment is parameterized by a distribution S over plaintext messages.

Experiment $\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda)$

$j \leftarrow 1, \mathcal{J} \leftarrow \emptyset$
$(n, t, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
$(pk, pkc, sk_1, \ldots, sk_n, ck, vk') \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t)$
$(D, \mathcal{I}, j^*, i^*, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{ct}(), \mathsf{corrupt}(\cdot)}(\mathsf{state}, pk, pkc)$
if $i^* \notin [n] \setminus \mathcal{J} \ \vee \ j^* \notin [j-1] \ \vee \ |\mathcal{J}| \geq t \ \vee \mathcal{I} \not\subseteq \mathcal{J}$ **then return** 0
$(\mathcal{I}', \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{state}, ck)$
if D is not exact admissible w.r.t. $(\mathcal{I}, t - |\mathcal{I}'| - 1)$ **then return** 0
if $\mathcal{I}' \not\subseteq \mathcal{I}$ **then return** 0
if $\mathsf{Verify}^D(pk, pkc, vk', \mathcal{I}' \cup \{i^*\}, c_{j^*}, \pi) = 1$ **then return** 1 **else return** 0

Oracle $\mathsf{corrupt}(i)$

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

Fig. 10. The confirmation integrity security experiment for a TDTC scheme $\mathcal{E}$ and an adversary $\mathcal{A}$. The experiment is parameterized by a distribution S over plaintext messages. The oracle ct is defined as in Fig. 9.

A TDTC scheme $\mathcal{TDTC}$

$\mathcal{TDTC}.\mathsf{Prove}^D(pk, \mathcal{I}, c^*, ck)$:

1. Parse ck as $\{sk_1, \ldots, sk_n\}$.
2. Compute decryption shares $(d_i, \pi_i) \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i, c^*)$ for all $i \in [n]$ and output $\{(d_i, \pi_i)\}_{i\in[n]}$.

$\mathcal{TDTC}.\mathsf{Verify}^D(pk, pkc, vk, \mathcal{I}, c^*, \pi)$:

1. Parse π as $\{(d_j, \pi_j)\}_{j\in[n]}$. If for some $j \in [n]$, $\mathcal{E}.\mathsf{ShareVf}(pk, vk, c^*, (d_j, \pi_j), j) = 0$, then output 0.
2. Compute the decryption of c^*, $m^* \leftarrow \mathcal{E}.\mathsf{Combine}(pkc, c^*, [t], \{(d_j, \pi_j)\}_{j\in[t]})$.
3. Partition the set $[n]\setminus\mathcal{I}$ into disjoint subsets each of size $k = t-|\mathcal{I}|$. Let us denote these sets by $\mathcal{S}_1, \ldots, \mathcal{S}_y$, where $y = \lceil \frac{n-|\mathcal{I}|}{k} \rceil$. If $n - |\mathcal{I}|$ does not divide k, then we wrap around for the last set $\mathcal{S}_y$.
4. For each $i \in [y]$, run $m_i \leftarrow D(\{(d_j, \pi_j)\}_{j\in\mathcal{S}_i})$. If for some $i \in [y]$, $m_i \neq m^*$, then output 0.
5. For each $i \in [y]$, let us denote $\mathcal{S}_i$ as $\{s_{i,1}, \ldots, s_{i,k}\}$. Then, for each $i \in [y], \ell \in [k], j \in \mathcal{I}$, let $\mathcal{S}_{i,\ell,j} \leftarrow \mathcal{S}_i \cup \{j\} \setminus \{s_{i,\ell}\}$ and run $m_{i,\ell,j} \leftarrow D(\{(d_w, \pi_w)\}_{w\in\mathcal{S}_{i,\ell,j}})$. If for any $i \in [y], \ell \in [k], j \in \mathcal{I}$, $m_{i,\ell,j}$ is equal to m^*, then output 0.
6. Otherwise, Output 1.

Fig. 11. Our TDTC scheme, denoted $\mathcal{TDTC}$, built from any robust threshold decryption scheme $\mathcal{E}$.

Theorem 8. *For every probabilistic polynomial time adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\text{conf}}_{\mathcal{TDTC},S,\mathcal{A}}(\lambda) \leq \frac{q_{\mathsf{ct}} \cdot (n+t)^2}{8 \cdot (1 - \mathsf{cp}(S))} \cdot \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{E},\mathcal{B}}(\lambda)$$

where $q_{\mathsf{ct}} = q_{\mathsf{ct}}(\lambda)$, $n = n(\lambda)$ and $t = t(\lambda)$ are an upper bound on the number of ct *calls $\mathcal{A}$ makes, and on the total number of parties and the threshold respectively.*

Proof. Let $\mathcal{I}$ be the set returned by $\mathcal{A}$, c_{j^*} be the j^*th ciphertext, and let π be the honestly generated proof in the game $\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda)$. Let $\mathsf{E}_1, \mathsf{E}_2, \mathsf{E}_3$ denote the events that the adversary wins because the checks in $\mathcal{TDTC}.\mathsf{Verify}(pk, pkc, vk, \mathcal{I}, c_{j^*}, \pi)$ fail at Lines 1 or 4 respectively. Then, by total probability, we have:

$$\begin{aligned}\mathsf{Adv}^{\text{conf}}_{\mathcal{TDTC},S}(\mathcal{A}) = &\Pr\left[\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_1\right] \\ &+ \Pr\left[\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_2\right] \\ &+ \Pr\left[\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3\right]\end{aligned}$$

Note that in the confirmation correctness game, the challenger honestly generates the proof π when given $pk, \mathcal{I}, c_{j^*}$ and the confirmation key which has the secret keys of all n parties. Hence, $\pi = \{(d_j, \pi_j)\}_{j\in[n]}$, where $(d_j, \pi_j) \leftarrow \mathcal{E}.\mathsf{Dec}(sk_j, c_{j^*})$ is a valid decryption share for party j for ciphertext c_{j^*}. Since the decryption shares are honestly generated, the validity check in $\mathcal{TDTC}.\mathsf{Verify}$ (Line 1) will go through (as stated in Section 2). This gives us,

$$\Pr\left[\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_1\right] = 0$$

Next, for any $i \in [y]$, the set $\mathcal{S}_i$ as defined in Line 3 trivially satisfies $\mathcal{S}_i \subseteq [n] \setminus \mathcal{I}$ and $|\mathcal{S}_i \cup \mathcal{I}| = t$. This means that, by the admissibility of the decoder (which is checked in Line 7 of the $\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda)$ game), the decoder must decrypt correctly when given shares of $\mathcal{S}_i$, for any $i \in [y]$. More formally, $D(\{(d_j, \pi_j)\}_{j\in\mathcal{S}_i}) = m_{j^*}$ for all $i \in [y]$.

Hence, either the game will output 0 at Line 7 or the check in Line 4 of the $\mathcal{TDTC}$.Verify protocol will also go through, meaning that,

$$\Pr\left[\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_2\right] = 0$$

This means that the only way for an adversary to win this game is to force the checks in Line 5 to fail. The following lemma shows that any adversary that does so, breaks semantic security of the underlying scheme $\mathcal{E}$, hence proving the theorem.

Lemma 3. *There exists an adversary $\mathcal{B}$ such that,*

$$\Pr\left[\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3\right] \leq \frac{q_{\mathsf{ct}} \cdot (n+t)^2}{8 \cdot (1 - \mathsf{cp}(S))} \cdot \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{E},S,\mathcal{B}}(\lambda)$$

where $q_{\mathsf{ct}} = q_{\mathsf{ct}}(\lambda)$, $n = n(\lambda)$ and $t = t(\lambda)$ are an upper bound on the number of $\mathsf{ct}()$ *queries by $\mathcal{A}$, on the total number of parties, and on the threshold respectively.*

Proof. Consider an adversary $\mathcal{B}$ playing the game $\mathbf{IND\text{-}CPA}_{\mathcal{E}}(\lambda)$. $\mathcal{B}$ invokes $\mathcal{A}$ and simulates the game $\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda)$ as follows. It gets (n,t) from $\mathcal{A}$ and forwards it to its challenger. It then gets pk, pkc from its challenger, and forwards (pk, pkc) to $\mathcal{A}$.

Let q_{ct} denote an upper bound on the number of $\mathsf{ct}()$ queries made by $\mathcal{A}$. $\mathcal{B}$ then samples distinct messages $m_0, m_1, \ldots, m_{q_{\mathsf{ct}}} \leftarrow\!\!\$\ S$ randomly from S. It guesses j^*; i.e. it samples $\hat{j} \leftarrow\!\!\$\ [q_{\mathsf{ct}}]$. If $m_0 = m_{\hat{j}}$, then $\mathcal{B}$ outputs a random bit $b' \leftarrow\!\!\$\ \{0,1\}$.

$\mathcal{B}$ forwards $(m_0, m_{\hat{j}})$ to its challenger, and gets back a ciphertext c. $\mathcal{B}$ also gets $\mathcal{I}$, the claimed effective set, from $\mathcal{A}$. $\mathcal{B}$ then queries its challenger for secret keys of all parties in $\mathcal{I}$, i.e. $\{sk_i \leftarrow \mathsf{corrupt}(i)\}_{i\in\mathcal{I}}$, and sends $\{sk_i\}_{i\in\mathcal{I}}$ to $\mathcal{A}$.

Next, $\mathcal{A}$ issues a sequence of queries. $\mathcal{B}$ initializes a counter $j_{\mathsf{ct}} = 0$ to keep track of the number of $\mathsf{ct}()$ queries so far. We now discuss how $\mathcal{B}$ responds to each one of $\mathcal{A}$'s queries:

- $\mathsf{ct}()$: $\mathcal{B}$ increments the counter $j_{\mathsf{ct}} \leftarrow j_{\mathsf{ct}} + 1$. If $j_{\mathsf{ct}} = \hat{j}$, then $\mathcal{B}$ outputs c, the ciphertext it received from its challenger. Otherwise, $\mathcal{B}$ encrypts $m_{j_{\mathsf{ct}}}$, i.e. $c_{j_{\mathsf{ct}}} \leftarrow \mathcal{E}.\mathsf{Enc}(pk, m_{j_{\mathsf{ct}}})$ and sends $c_{j_{\mathsf{ct}}}$ to $\mathcal{A}$.

The adversary $\mathcal{A}$ eventually outputs a decoder D and an index j^*. If $j^* \neq \hat{j}$, then $\mathcal{B}$ aborts. And if any of the checks in Line 6 or 7 fail, the game will output 0 and $\mathcal{B}$ will output a random bit $b' \leftarrow\!\!\$\ \{0,1\}$.

Next, if the event E_3 occurs, there exists some $i \in [y], \ell \in [k], z \in \mathcal{I}$ such that the decoder decrypts correctly when given the decryption shares for the set $\mathcal{S}_i \cup \{z\} \setminus \{s_{i,\ell}\}$, where $y, k, \mathcal{S}_i$ and $s_{i,\ell}$ are as defined in Line 3 of the $\mathcal{TDTC}$.Verify algorithm.

$\mathcal{B}$ samples $\hat{i} \leftarrow\!\!\$\ [y]$, $\hat{z} \leftarrow\!\!\$\ \mathcal{I}$ and $\hat{\ell} \leftarrow\!\!\$\ [k]$ uniformly randomly, as its guesses for i, z and ℓ respectively.

Then, $\mathcal{B}$ queries its challenger for secret keys of all the parties in $\mathcal{S}_{\hat{i}} \cup \mathcal{I} \setminus (\mathcal{J} \cup \{s_{\hat{i},\hat{\ell}}\})$, i.e. it calls $sk_i \leftarrow \mathsf{corrupt}(i)$ for all $i \in \mathcal{S}_{\hat{i}} \cup \mathcal{I} \setminus (\mathcal{J} \cup \{s_{\hat{i},\hat{\ell}}\})$.

It then runs the decoder D with the decryption shares for the set: $\mathcal{S}_{\hat{i},\hat{z},\hat{\ell}} \leftarrow \mathcal{S}_{\hat{i}} \cup \{\hat{z}\} \setminus \{s_{\hat{i},\hat{\ell}}\}$. More formally, it runs $m \leftarrow D(\{(d_j, \pi_j)\}_{j\in\mathcal{S}_{\hat{i},\hat{z},\hat{\ell}}})$, where $(d_j, \pi_j) \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i, c)$ for all $j \in \mathcal{S}_{\hat{i},\hat{z},\hat{\ell}}$.

If $m = m_0$, $\mathcal{B}$ outputs 0 to its challenger. If $m = m_{j^*}$, $\mathcal{B}$ outputs 1 to its challenger. Otherwise, it outputs a random bit $b' \leftarrow\!\!\$\ \{0,1\}$.

Observe that $\mathcal{B}$ wins its ind-cpa game if $\mathcal{B}$ does not abort, and guesses $\hat{i}, \hat{\ell}, \hat{z}$ correctly, and if $\mathcal{A}$ wins the **ExpConfirm** game (and the event E_3 occurs). This is because, first, $\mathcal{A}$ wins the game only if, the check on Line 5 fails for some i, z, ℓ, i.e. if the decoder decrypts correctly for some i, z, ℓ. This means that if $\mathcal{B}$ guesses $\hat{i}, \hat{z}, \hat{\ell}$ correctly, then its response to its challenger would be correct. Note that if it guesses incorrectly, then $\mathcal{B}$ can either win or lose the game, depending upon the decoder's behavior.

Secondly, the total number of $\mathsf{corrupt}()$ queries that $\mathcal{B}$ made is less than t, since it only queried the secret keys for the set $\mathcal{I}$ and $\mathcal{S}_{\hat{i}} \setminus \{s_{\hat{i},\hat{\ell}}\}$, and by definition of the sets $\mathcal{S}_i$, $|\mathcal{I} \cup \mathcal{S}_{\hat{i}} \setminus \{s_{\hat{i},\hat{\ell}}\}| = t - 1$.

Let Abort denote the event in which $\mathcal{B}$ aborts. Let E_m be the event that $m_0 = m_{\hat{j}}$. Let $\mathsf{E}_{i,z,l}$ be the event that $\mathcal{B}$ correctly guessed $\hat{i}, \hat{z}, \hat{l}$, i.e. it guessed the set $\mathcal{S}_{\hat{i}}$ and the elements $z \in \mathcal{I}$, $s_{\hat{i},\hat{l}}$ such that the decoder does indeed decrypt correctly when input shares of $\mathcal{S}_{\hat{i},\hat{z},\hat{l}}$. So we get that,

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{E},S}(\mathcal{B}) \geq \Pr\left[(\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3) \wedge \overline{\mathsf{Abort}} \wedge \mathsf{E}_{i,z,l} \wedge \overline{\mathsf{E}}_m\right]$$

We will now analyse the probability of the events Abort, E_m and $\mathsf{E}_{i,z,l}$. We have that $\Pr[\mathsf{E}_{i,z,l}] \geq \frac{1}{k} \cdot \frac{1}{y} \cdot \frac{1}{|\mathcal{I}|} \geq \frac{1}{(n+t-2|\mathcal{I}|)\cdot|\mathcal{I}|} \geq \frac{8}{(n+t)^2}$, since $k = t - |\mathcal{I}|$, $y = \lceil \frac{n-|\mathcal{I}|}{k} \rceil$. Let E_j denote the event that $\mathcal{B}$ guessed $\hat{j}$ correctly, i.e. $\hat{j} = j^*$. We have that $\Pr[\mathsf{E}_j] \geq \frac{1}{q_{\mathsf{ct}}}$. Lastly, $\Pr[\mathsf{E}_m] = \mathsf{cp}(S)$. We get,

$$\begin{aligned}
\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{E},S}(\mathcal{B}) &\geq \Pr\left[(\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3) \wedge \overline{\mathsf{Abort}} \wedge \mathsf{E}_{i,z,l} \wedge \overline{\mathsf{E}}_m\right] \\
&\geq \Pr\left[(\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3)\right] \cdot \Pr[\overline{\mathsf{Abort}} \wedge \mathsf{E}_{i,z,l} \wedge \overline{\mathsf{E}}_m] \\
&\geq \Pr\left[(\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3)\right] \cdot \Pr[\mathsf{E}_j \wedge \mathsf{E}_{i,z,l} \wedge \overline{\mathsf{E}}_m] \\
&\geq \Pr\left[(\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3)\right] \cdot \Pr[\mathsf{E}_j] \cdot \Pr[\mathsf{E}_{i,z,l}] \cdot \Pr[\overline{\mathsf{E}}_m] \\
&\geq \Pr\left[(\mathbf{ExpConfirm}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \mathsf{E}_3)\right] \cdot \frac{1}{q_{\mathsf{ct}}} \cdot \frac{8}{(n+t)^2} \cdot (1 - \mathsf{cp}(S))
\end{aligned}$$

This proves the lemma.

We now prove that it is hard for any probabilistic polynomial time adversary to frame an honest party for the scheme $\mathcal{TDTC}$.

Theorem 9. *For every probabilistic polynomial time adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}_2$ such that,*

$$\mathsf{Adv}^{\text{frame}}_{\mathcal{TDTC},S,\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{dc}}_{\mathcal{E},\mathcal{B}_2}(\lambda)$$

Proof. Observe that $\mathcal{A}$ can win only if $\mathcal{TDTC}.\mathsf{Verify}(pk, pkc, vk, (\mathcal{I}' \cup \{i^*\}), c_{j^*}, \pi) = 1$, where c_{j^*} and π refer to the ciphertext that the decoder decrypts, and the proof provided by $\mathcal{A}$.

Let $\mathsf{E}_1, \mathsf{E}_2$ denote the events that the checks in $\mathcal{TDTC}.\mathsf{Verify}(pk, pkc, vk, (\mathcal{I}' \cup \{i^*\}), c_{j^*}, \pi)$ fail at Lines 1 and 4 respectively. Next, let $\{(d_i, \pi_i) \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i, c_{j^*})\}_{i\in[n]}$ be the honest decryption shares and their corresponding robustness proofs for the ciphertext c_{j^*}. Then, let E_{r} denote the event that the proof π sent by $\mathcal{A}$ contains the honest decryption shares for all n parties : $\{d_i\}_{i\in[n]}$. Observe that the adversary loses the game if either of the events E_1 or E_2 occur. Hence, by total probability, we have that,

$$\begin{aligned}\mathsf{Adv}^{\text{frame}}_{\mathcal{TDTC},S}(\mathcal{A}) &= \Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_2\right] \\ &= \Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_2 \wedge \mathsf{E_r}\right] \\ &\quad + \Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_2 \wedge \overline{\mathsf{E}}_{\mathsf{r}}\right]\end{aligned}$$

In other words, $\mathcal{A}$ can win only if it produces a proof containing decryption shares that pass the robustness check for all parties in $[n]$, and if the decoder runs correctly when given shares of parties in $\mathcal{S}_i$ for all $i \in [y]$. The following lemmas together prove the theorem. Intuitively, they show that such an adversary can be used to break the robustness of the underlying encryption scheme $\mathcal{E}$.

Lemma 4. *There exists an adversary $\mathcal{B}_2$ such that,*

$$\Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_2 \wedge \overline{\mathsf{E}}_{\mathsf{r}}\right] \leq \mathsf{Adv}^{\text{dc}}_{\mathcal{E},\mathcal{B}_2}(\lambda)$$

Proof. Consider an adversary $\mathcal{B}_2$ playing the game $\mathbf{DC}_{\mathcal{E}}(\lambda)$. $\mathcal{B}_2$ invokes $\mathcal{A}$ and simulates the game $\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda)$ as follows. It gets (n, t) from $\mathcal{A}$ and forwards it to its challenger. It then gets $(pk, pkc, sk_1, \ldots, sk_n)$ from its challenger. It forwards (pk, pkc) to $\mathcal{A}$.

Let q_{ct} denote an upper bound on the number of $\mathsf{ct}()$ queries made by $\mathcal{A}$. $\mathcal{B}_2$ maintains a counter j_{ct} and a mapping $C : [q_{\mathsf{ct}}] \to \mathcal{M} \times \mathcal{C}$ to store metadata for responding to $\mathsf{ct}(\cdot)$ queries. Here, $\mathcal{C}$ is the space of the ciphertexts.

$\mathcal{A}$ issues a sequence of queries. We now discuss how $\mathcal{B}_2$ responds to each one of $\mathcal{A}$'s queries:

- $\mathsf{ct}()$: $\mathcal{B}_2$ increments the counter $j_{\mathsf{ct}} \leftarrow j_{\mathsf{ct}} + 1$. It then samples a message $m \leftarrow\$ \mathcal{M}$ and encrypts it $c \leftarrow \mathcal{E}.\mathsf{Enc}(pk, m)$. $\mathcal{B}_2$ stores $C(j_{\mathsf{ct}}) \leftarrow (m, c)$ and outputs c.
- $\mathsf{corrupt}(i)$: $\mathcal{B}_2$ sends sk_i to $\mathcal{A}$.

$\mathcal{A}$ eventually outputs a decoder D, a set $\mathcal{I}$, an index $j^* \in [q_{\mathsf{ct}}]$ and an innocent party $i^* \in [n] \setminus \mathcal{J}$. Let $(m^*, c^*) \leftarrow C(j^*)$ be the message and the ciphertext corresponding to the j^*th $\mathsf{ct}()$ query.

$\mathcal{B}_2$ then sends $\{sk_i\}_{i \in [n]}$ to $\mathcal{A}$, and receives a set $\mathcal{I}' \cup \{i^*\}$ and a proof $\pi = \{(\hat{d}_i, \hat{\pi}_i)\}_{i \in [n]}$.

$\mathcal{B}_2$ computes honest decryption shares and robustness proofs for all parties in $[n]$, i.e. $(d_i, \pi_i) \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i, c^*)$ Let ℓ^* be the smallest index in $[n]$ such that $d_{\ell^*} \neq \hat{d}_{\ell^*}$. Note that, in the case of events $\overline{\mathsf{E}}_1$ and $\overline{\mathsf{E}}_r$, there must be one such index ℓ^*, and, the shares in π must all be valid, i.e. $\mathsf{ShareVf}(pk, vk, c^*, (\hat{d}_{\ell^*}, \hat{\pi}_{\ell^*}), \ell^*) = 1$. By the correctness of the encryption scheme, we have that $\mathsf{ShareVf}(pk, vk, c^*, (d_{\ell^*}, \pi_{\ell^*}), \ell^*) = 1$. $\mathcal{B}_2$ returns $(c^*, (d_{\ell^*}, \pi_{\ell^*}), (\hat{d}_{\ell^*}, \hat{\pi}_{\ell^*}), \ell^*)$ to its challenger. As per the discussion above, if $\mathcal{A}$ wins the game $\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda)$, and if the events $\overline{\mathsf{E}}_1$ and $\overline{\mathsf{E}}_r$ occur, then, $\mathcal{B}_2$ wins the game $\mathbf{DC}_{\mathcal{E}}(\lambda)$. This gives us,

$$\begin{aligned}\mathsf{Adv}^{\text{dc}}_{\mathcal{E},\mathcal{B}_2}(\lambda) &= \Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_{\mathsf{r}}\right] \\ &\geq \Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_2 \wedge \overline{\mathsf{E}}_{\mathsf{r}}\right]\end{aligned}$$

This proves the lemma.

Lemma 5.

$$\Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_2 \wedge \mathsf{E_r}\right] = 0$$

Proof. For any adversary $\mathcal{A}$, in the events $\overline{\mathsf{E}}_1, \overline{\mathsf{E}}_2$ and E_r, we have that,

– E_r implies that the proof π sent by $\mathcal{A}$ is $\{(d_i, \hat{\pi}_i)\}_{i\in[n]}$, where $(d_i, \pi_i) \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i, c_{j^*})$ for all $i \in [n]$ (π_i may or may not be equal to $\hat{\pi}_i$).
– Then, $\overline{\mathsf{E}}_2$ implies that all the checks in Line 4 will go through. This implies that the decoder decrypts correctly when given decryption shares and robustness proofs of $\mathcal{S}_i$ for all $i \in [y]$.
– By the admissibility check (Line 7), we know that $\mathcal{A}$ can win only if $D = D_{\mathcal{I}}$. This, combined with the fact that $D(\{(d_i, \hat{\pi}_i)\}_{i\in\mathcal{S}_1}) = m$ implies that, $|\mathcal{S}_1 \cup \mathcal{I}| \geq t$.
– Next, consider the set $\mathcal{S}_{1,\ell} = \mathcal{S}_1 \cup \{i^*\} \setminus \{s_{1,\ell}\}$, for any $\ell \in [k]$, where $\mathcal{S}_1 = \{s_{1,1}, \ldots, s_{1,k}\}$. Since, $i^* \notin \mathcal{I}$, adding i^* and removing $s_{1,\ell}$ cannot reduce the size of intersection with $\mathcal{I}$. More formally,

$$|\mathcal{S}_1 \cup \mathcal{I}| \geq t \ \wedge \ i^* \notin \mathcal{I} \implies |\mathcal{S}_1 \cup \{i^*\} \setminus \{s_{1,\ell}\} \cup \mathcal{I}| \geq t$$

By the admissibility of the decoder, this implies that the decoder must decrypt correctly when given decryption shares of $\mathcal{S}_{1,\ell}$ for any $\ell \in [k]$. This means that the checks in Line 5 in the Verify procedure will fail for $i = 1, i^* \in (\mathcal{I}' \cup \{i^*\}), \ell \in [k]$, meaning that the adversary cannot win. More formally,

$$\Pr\left[\mathbf{ExpFrame}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1 \wedge \overline{\mathsf{E}}_1 \wedge \overline{\mathsf{E}}_2 \wedge \mathsf{E}_\mathsf{r}\right] = 0$$

This proves the lemma.

Handling probabilistic decoders. We presented our algorithms for admissible decoders that are deterministic and perfect. Note they can be easily generalized to handle probabilistic decoders, which decrypt correctly with a certain non-negligible probability.

First, we extend the definition of admissible decoders to probabilistic decoders. We focus on general encryption schemes for the definition below, but if a robust threshold encryption scheme is used, the decoder also takes the robustness proofs as input, along with the decryption shares.

Definition 13. *We say that a decoder $D_{\mathcal{I}}$ for a cipher text $c \xleftarrow{\$} \mathsf{Enc}(pk, m)$ is admissible with respect to a set of parties $\mathcal{I} \subseteq [n]$, if the decoder decrypts with non-negligible probability if and only if, it is given at least $t - |\mathcal{I}|$ valid decryption shares for parties in $[n] \setminus \mathcal{I}$. Namely, there exists non-negligible functions $\epsilon = \epsilon(\lambda)$ and a polynomial $q = q(\lambda)$*

$$\forall \mathcal{S} \subseteq [n], |\mathcal{S} \cup \mathcal{I}| \geq t \implies \Pr\left[D_{\mathcal{I}}(\{d_j\}_{j\in\mathcal{S}}) = m\right] \geq \epsilon \tag{10}$$

$$\forall \mathcal{S} \subseteq [n], |\mathcal{S} \cup \mathcal{I}| < t \implies \Pr\left[D_{\mathcal{I}}(\{d_j\}_{j\in\mathcal{S}}) = m\right] \leq \epsilon - 1/q(\lambda) \tag{11}$$

Intuitively, Eq. (11) enforces the fact that $\mathcal{I}$ is precisely the subset underlying the decoder D. Moreover, if the adversary can only corrupt the parties in $\mathcal{I}$, this follows from the semantic security of the scheme. In this case, we should think of $\epsilon - 1/q$ as negligible, due to the semantic security of the scheme. Hence, in particular, $1/q \geq \epsilon/2$ and hence non-negligible. Allowing arbitrary polynomials q makes our definition more general.

To extend our $\mathcal{TDTC}$ scheme for a probabilistic admissible decoder, the Verify procedure now takes ϵ and $q(\lambda)$ as input. Additionally, we modify Steps 4 and 5 as follows:

– (Step 4): For each $i \in [y]$, run the decoder $W = \Omega(\lambda \cdot (q(\lambda))^2)$ times on decryption shares of $\mathcal{S}_i$. Let $m_{i,r} \leftarrow\$ D(\{(d_j, \pi_j)\}_{j\in\mathcal{S}_i})$ for all $r \in [W]$. Let $\mathsf{ct}_i = \Sigma_{r\in W} \mathbb{1}[m_{i,r} = m^*]$. If for some $i \in [y]$, $\frac{\mathsf{ct}_i}{W} < \epsilon - \frac{1}{2q(\lambda)}$, then output 0.

– (Step 5): For each $i \in [y], \ell \in [k], j \in \mathcal{I}$, run the decoder W times on shares of $\mathcal{S}_{i,\ell,j}$. Let $m_{i,\ell,j,r} \leftarrow\$ D(\{(d_s, \pi_s)\}_{s \in \mathcal{S}_{i,\ell,j}})$ for all $r \in [W]$, and let $\mathsf{ct}_{i,\ell,j} = \Sigma_{r \in W} \mathbb{1}[m_{i,\ell,j,r} = m^*]$. If for some $i \in [y], \ell \in [k], j \in \mathcal{I}$, $\frac{\mathsf{ct}_{i,\ell,j}}{W} > \epsilon - \frac{1}{2q(\lambda)}$, then output 0.

We claim that the above algorithm satisfies confirmation correctness and integrity. Let $p_{\mathcal{S}}$ be the probability that the decoder decrypts correctly when given decryption shares of parties in $\mathcal{S} \subseteq [n]$. Then, running the decoder W times with shares of $\mathcal{S}$ allows us to get a good estimate of this probability. More formally, let $\hat{p}_{\mathcal{S}}$ denote $\frac{\mathsf{ct}_{\mathcal{S}}}{W}$, where $\mathsf{ct}_{\mathcal{S}}$ is the number of times the decoder decrypts correctly when run W times with shares of $\mathcal{S}$. Then, by Chernoff bound, we have that $\Pr[|\hat{p}_{\mathcal{S}} - p_{\mathcal{S}}| > 1/4q(\lambda)] < 2e^{-\Omega(\lambda)}$. This means that with high probability, $\hat{p}_{\mathcal{S}} < \epsilon - 1/2q(\lambda)$ implies that $p_{\mathcal{S}}$ must be less than $\epsilon - 1/q(\lambda)$, and that $|\mathcal{S} \cup \mathcal{I}| < t$ (by admissibility of the decoder), and vice versa. Hence, confirmation and integrity can be argued similarly to the proofs for Theorems 8 and 9.

6.3 Tracing Small Traitor Coalitions

In this section, we define traitor tracing in the small traitor coalition setting, where the number of corruptions f is less than the threshold t. We start with definitions and a construction for deterministic admissible decoders, and discuss how to generalize to probabilistic admissible decoders later in this section.

A threshold decryption with traitor tracing (or TDTT for short) is a threshold decryption with an additional PPT tracing procedure Trace, which takes as input the public key pk, the combiner public key pkc, the tracing key tk which is an additional output of KeyGen, and the ciphertext c along with oracle access to a decoder D that can decrypt c. It outputs a subset $\mathcal{J} \subseteq [n]$.

A secure threshold decryption with traitor tracing must satisfy **tracing correctness**. This notion is defined similarly to the large coalition case, but considers universal admissible decoders that take in decryption shares. The definition is also simplified compared to the large coalition case, since we focus on ciphertext-specific deterministic decoders. We will discuss how to generalize the definition and our construction to probabilistic decoders later in this section. We formally describe the security experiment in Fig. 12.

Experiment $\mathbf{ExpTraceSmall}_{\mathcal{A},\mathcal{E},\epsilon}(\lambda)$

$\mathcal{J} \leftarrow \emptyset$
$(n, t, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
$(pk, pkc, sk_1, \ldots, sk_n, tk) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, n, t)$
$(D, \mathcal{I}_{\mathsf{eff}}, j^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{corrupt}(\cdot),\mathsf{ct}(\cdot)}(\mathsf{state}, pk, pkc)$ // output a decoder alg. D
if $|\mathcal{J}| \geq t \vee \mathcal{I}_{\mathsf{eff}} \not\subseteq \mathcal{J} \vee j^* \notin [j-1]$ **then return** 0
if D is not universal admissible w.r.t. $\mathcal{I}_{\mathsf{eff}}$ **then return** 0
$\mathcal{J}' \xleftarrow{\$} \mathsf{Trace}^{D(\cdot)}(pk, pkc, c_{j^*}, tk)$ // trace decoder
return $\mathcal{J}' \neq \mathcal{I}_{\mathsf{eff}}$

Fig. 12. The small coalition tracing experiment for a threshold decryption scheme $\mathcal{E}$ and an adversary $\mathcal{A}$. The oracles $\mathsf{corrupt}$ and ct are defined as in Fig. 1 and Fig. 9, respectively.

Definition 14. *We say that $\mathcal{TDTT}$ has tracing correctness if for every probabilistic polynomial time adversary $\mathcal{A}$, the following function is negligible in λ:*

$$\mathsf{Adv}^{\mathrm{tc}}_{\mathcal{A},\mathcal{E},S}(\lambda) = \Pr\left[\mathbf{ExpTraceSmall}_{\mathcal{A},\mathcal{E},S}(\lambda) = 1\right]$$

We now build a generic traitor tracing system for universal admissible decoders that accept any $1 \leq x \leq t$ number of shares. We do so from any semantically-secure threshold decryption scheme. The algorithms $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Combine}$ remain the same as the underlying encryption scheme. The Trace procedure is described in Fig. 13, and it takes the secret keys of all n parties as the tracing key, i.e. $tk \leftarrow \{sk_i\}_{i\in[n]}$.

Theorem 10 below proves tracing correctness.

$\mathcal{TDTT}$: A threshold scheme with small coalition tracing

$\mathcal{TDTT}.\mathsf{Trace}^D(pk, pkc, c^*, tk)$:

1. Parse tk as $\{sk_1, \ldots, sk_n\}$.
2. Split $[n]$ into disjoint subsets of size t, namely $\mathcal{T}_1, \ldots, \mathcal{T}_z$, where $\mathcal{T}_j = \{t \cdot (j-1)+1, \ldots, tj\}$, where $z = \frac{n}{t}$. We assume n divides t for simplicity, but this can be generalized by wrapping around for the last subset $\mathcal{T}_z$.
3. Compute decryption shares for all parties, i.e. $d_j \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i, c^*)$ for all $j \in [n]$. Compute the decryption of the ciphertext; $m^* \leftarrow \mathcal{E}.\mathsf{Combine}(pkc, c^*, [t], \{d_i\}_{i\in[t]})$.
4. Let $\mathcal{I}^* = \emptyset$. For $i = 1, \ldots, z$ do:
   (a) Set $\mathcal{J}_i \leftarrow \mathcal{T}_i$, and $\mathcal{H}_i \leftarrow \perp$.
   (b) Let $\mathcal{J}_i = \{j_1, \ldots, j_y\}$, where $y = |\mathcal{J}_i|$.
   (c) If $D(\{d_v\}_{v\in\mathcal{J}_i}) = m^*$, then,
      i. For each $r \in [y+1]$, run the decoder with decryption shares of $\{j_r, \ldots, j_y\}$, i.e. $m_r \leftarrow D(\{d_v\}_{v\in\{j_r,\ldots,j_y\}})$.
      ii. Let $w \in [y]$ be the smallest value such that $m_w = m^*$ but $m_{w+1} \neq m^*$. Abort if there is no such w.
      iii. Then, set $\mathcal{H}_i \leftarrow \mathcal{H}_i \cup \{j_w\}$, and $\mathcal{J}_i \leftarrow \mathcal{J}_i \setminus \{j_w\}$. Run step 4b with the updated $\mathcal{J}_i, \mathcal{H}_i$ sets.
   (d) Otherwise, if $D(\{d_v\}_{v\in\mathcal{J}_i}) \neq m^*$, then,
      i. Let $\mathcal{H}_i = \{h_1, \ldots, h_u\}$ where $u = |\mathcal{H}_i|$.
      ii. For each $r \in [u]$, run the decoder with decryption shares of $\mathcal{J}_i \cup \{h_1, \ldots, h_r\}$, i.e. set $\hat{m}_r \leftarrow D(\{d_v\}_{v\in\mathcal{J}_i\cup\{h_1,\ldots,h_r\}})$.
      iii. Let $\hat{w}$ be the first (smallest) value such that $\hat{m}_{\hat{w}} = m^*$. Abort if there is no such value $\hat{w}$.
      iv. If the decoder works when given shares of $\{h_1, \ldots, h_{\hat{w}}\}$, i.e. if $D(\{d_v\}_{v\in\{h_1,\ldots,h_{\hat{w}}\}}) = m^*$, then all parties in $\mathcal{J}_i$ are traitors, so we set $\mathcal{I}^* \leftarrow \mathcal{I}^* \cup \mathcal{J}_i$.
      v. Otherwise, if $D(\{d_v\}_{v\in\{h_1,\ldots,h_{\hat{w}}\}}) \neq m^*$, then, for every $r \in [y+1]$, run the decoder with decryption shares of $\{j_r, \ldots, j_y\} \cup \{h_1, \ldots, h_{\hat{w}}\}$, i.e. $m'_r \leftarrow D(\{d_v\}_{v\in\{j_r,\ldots,j_y\}\cup\{h_1,\ldots,h_{\hat{w}}\}})$. Let w' be the smallest value such that $m'_{w'} = m^*$ but $m'_{w'+1} \neq m^*$. Abort if there is no such value. Otherwise, set $\mathcal{J}_i \leftarrow \mathcal{J}_i \setminus \{j_{w'}\}$ and $\mathcal{H}_i \leftarrow \mathcal{H}_i \cup \{j_{w'}\}$. Run step 4b with these updated values.
5. Output $\mathcal{I}^*$.

Fig. 13. A threshold decryption scheme with tracing of small traitor coalitions, denoted $\mathcal{TDTT}$, built from any robust threshold decryption scheme.

Theorem 10. *Let $\mathcal{TDTT}$ be a threshold decryption scheme with traitor tracing. Then, for every probabilistic polynomial time adversary $\mathcal{A}$, we have that,*

$$\mathsf{Adv}^{\mathrm{tc}}_{\mathcal{A},\mathcal{E},S}(\lambda) = 0$$

Before proving the theorem, we note that the advantage of the adversary is 0 due to our assumption that the decoder is deterministic and perfect (i.e., it always decrypts with at least t shares, but never decrypts with less). Looking ahead, when lifting this restriction, the advantage will become negligible.

Proof. First, observe that the adversary can only win the game if it makes less than t $\mathsf{corrupt}(\cdot)$ calls, outputs $\mathcal{I}_{\mathsf{eff}} \subseteq \mathcal{J}$ and $j^* \in [j-1]$. Next, Line 6 ensures that the adversary will lose the game if it outputs a non-admissible decoder, or if it outputs any set $\mathcal{I}_{\mathsf{eff}}$ that is not indeed the effective traitor set of the decoder D. More formally, D must be a universal admissible decoder with respect to $\mathcal{I}_{\mathsf{eff}}$, for the adversary to win. Given all of the above, we prove that the Trace algorithm will be able to trace the decoder to the set $\mathcal{I}_{\mathsf{eff}}$.

First, observe that, during the algorithm, the following holds for all $i \in [z]$ at all times:

$$\mathcal{J}_i \cup \mathcal{H}_i = \mathcal{T}_i \tag{12}$$

$$\mathcal{J}_i \cap \mathcal{H}_i = \phi \tag{13}$$

This can be seen by induction, based on the fact that, (a) at the beginning of the algorithm, $\mathcal{J}_i = \mathcal{T}_i$ and $\mathcal{H}_i = \perp$, and (b) every time these sets are updated, an element v is removed from $\mathcal{J}_i$, and it is added to $\mathcal{H}_i$. This also means that, each time step 4b is run, either the algorithm finishes execution or, the size of $\mathcal{J}_i$ reduces by one.

Lemma 6. *For any $\mathcal{TDTT}$ scheme, and any admissible decoder D, the* Trace *algorithm does not abort for any $i \in [z]$.*

Proof. Observe that, for any $i \in [z]$, whenever the algorithm reaches Step 4c, it holds that the decoder outputs m^* when given shares of $\mathcal{J}_i$ as input. For an admissible decoder, this implies that,

$$|\mathcal{J}_i \cup \mathcal{I}_{\mathsf{eff}}| \geq t$$

At the same time, the decoder does not output m^* when given no shares as input, since $|\mathcal{I}_{\mathsf{eff}}| < t$, i.e. $|\perp \cup \mathcal{I}_{\mathsf{eff}}| < t$.

Let us use $\mathcal{J}_{i,r}$ to denote the set $\{j_r, \ldots, j_y\}$ for $r \in [y+1]$, and let $I(\mathcal{S})$ be defined as $|\mathcal{S} \cup \mathcal{I}_{\mathsf{eff}}|$ for any set $\mathcal{S} \subseteq [n]$. By definition, $I(\mathcal{J}_{i,r})$ is a monotonic, non-increasing function with respect to r, specifically, we know that $I(\mathcal{J}_{i,r}) - 1 \leq I(\mathcal{J}_{i,r+1}) \leq I(\mathcal{J}_{i,r})$. We also have that $I(\mathcal{J}_{i,1}) = I(\mathcal{J}_i) \geq t$ and $I(\mathcal{J}_{i,y+1}) = I(\mathcal{I}_{\mathsf{eff}}) < t$. This means that there must exist a value $v \in [y]$ such that $I(\mathcal{J}_{i,v}) = t$ and $I(\mathcal{J}_{i,v+1}) = t - 1$. For this v, m_v will be m^* and m_{v+1} will be $\neq m^*$, since D is an admissible decoder. Hence, the algorithm will never abort at Step 4(c)ii.

Next, whenever the algorithm reaches Step 4d, we have that $D(\{d_v\}_{v \in \mathcal{J}_i}) \neq m^*$, implying that $I(\mathcal{J}_i) < t$. Let us define $\hat{\mathcal{J}}_{i,r}$ to be the set $\mathcal{J}_i \cup \{h_1, \ldots, h_r\}$, for $r \in [u] \cup \{0\}$. Again, $I(\hat{\mathcal{J}}_{i,r})$ is a monotonic, non-decreasing function with respect to r, specifically, $I(\hat{\mathcal{J}}_{i,r}) \leq I(\hat{\mathcal{J}}_{i,r+1}) \leq I(\hat{\mathcal{J}}_{i,r}) + 1$. Lastly, since we know from Equation 12 that $|\mathcal{J}_i \cup \mathcal{H}_i| = |\mathcal{T}_i| = t$, we have that $I(\hat{\mathcal{J}}_{i,u}) = I(\mathcal{T}_i) \geq t$. Combined with the fact that $I(\hat{\mathcal{J}}_{i,0}) = I(\mathcal{J}_i) < t$, we get that there must exist a $\hat{v} \in [u]$ such that $I(\hat{\mathcal{J}}_{i,\hat{v}-1}) = t - 1$ and $I(\hat{\mathcal{J}}_{i,\hat{v}}) = t$. This means that $\hat{m}_{\hat{v}}$ will be m^*, and hence, the algorithm will not abort at Step 4(d)iii.

Next, there are two possibilities:

- The decoder outputs m^* when given shares of $\{h_1, \ldots, h_{\hat{w}}\}$. In this case, the iterative algorithm finishes for $\mathcal{T}_i$.

– The decoder does not output m^* when given shares of $\{h_1, \ldots, h_{\hat{w}}\}$, meaning that $I(\{h_1, \ldots, h_{\hat{w}}\}) < t$. For this case, let us define $\mathcal{J}'_{i,r}$ to be the set $\{j_r, \ldots, j_y\} \cup \{h_1, \ldots, h_{\hat{w}}\}$ for $r \in [y+1]$. We have that $I(\mathcal{J}'_{i,r})$ is a monotonic, non-increasing function with respect to r, and $I(\mathcal{J}'_{i,y+1}) = I(\{h_1, \ldots, h_{\hat{w}}\}) < t$, and $I(\mathcal{J}'_{i,1}) = I(\hat{\mathcal{J}}_{i,\hat{w}}) \geq t$. Hence, there must exist a value $v' \in [y]$ such that $I(\mathcal{J}'_{i,v'}) = t$ but $I(\mathcal{J}'_{i,v'+1}) = t - 1$, and the algorithm will not abort at Step 4(d)v.

Lemma 7. *For each i, at the end of the iterative algorithm, we have,*

$$\mathcal{J}_i \subseteq \mathcal{I}_{\mathsf{eff}}$$

Proof. We will prove this by contradiction.

For each i, when the algorithm ends (at Step 4(d)iv), we have that,

$$D(\{d_v\}_{v \in \mathcal{J}_i}) \neq m^*$$

By the definition of an admissible decoder, this implies that

$$|\mathcal{J}_i \cup \mathcal{I}_{\mathsf{eff}}| < t$$

We also have that, $\hat{m}_{\hat{w}} = m^*$ and $\hat{m}_{\hat{w}-1} \neq m^*$. These combined with the definition of an admissible decoder imply the following:

$$|\mathcal{J}_i \cup \{h_1, \ldots, h_{\hat{w}-1}\} \cup \mathcal{I}_{\mathsf{eff}}| < t$$

$$|\mathcal{J}_i \cup \{h_1, \ldots, h_{\hat{w}}\} \cup \mathcal{I}_{\mathsf{eff}}| \geq t$$

Since adding just one element makes the size of the union cross the threshold, we get that

$$|\mathcal{J}_i \cup \{h_1, \ldots, h_{\hat{w}}\} \cup \mathcal{I}_{\mathsf{eff}}| = t \tag{14}$$

Lastly, we also have that the decoder decrypts correctly when given decryption shares of $\{h_1, \ldots, h_{\hat{w}}\}$, which implies that,

$$|\{h_1, \ldots, h_{\hat{w}}\} \cup \mathcal{I}_{\mathsf{eff}}| \geq t \tag{15}$$

Note that $h_j \in \mathcal{H}_i$ for all $j \in [\hat{w}]$. Then, Equations 15 and 14 combined with the fact that $\mathcal{J}_i \cup \mathcal{H}_i = \phi$ can only hold together if $\mathcal{J}_i \subseteq \mathcal{I}_{\mathsf{eff}}$. This can be seen by contradiction; if there were even a single element $u \in \mathcal{J}_i \setminus \mathcal{I}_{\mathsf{eff}}$, then, $|\mathcal{J}_i \cup \{h_1, \ldots, h_{\hat{w}}\} \cup \mathcal{I}_{\mathsf{eff}}|$ would have to be strictly larger than $|\{h_1, \ldots, h_{\hat{w}}\} \cup \mathcal{I}_{\mathsf{eff}}|$.

Lemma 8. *For any party h that is added to $\mathcal{H}_i$, $h \notin \mathcal{I}_{\mathsf{eff}}$.*

Proof. Let $\mathcal{I}_{\mathsf{eff}}$ denote the effective traitor set of the decoder D, as defined in Definition 10. There are two steps in the algorithm where a party might be added to $\mathcal{H}_i$. We will consider them one by one:

– Step 4c. Here, a party w is added to $\mathcal{H}_i$, if $m_w = m^*$ and $m_{w+1} \neq m^*$. By the definition of an admissible decoder, we have that,

$$\{j_w, \ldots, j_y\} \cup \mathcal{I}_{\mathsf{eff}} \geq t$$

and,

$$\{j_{w+1}, \ldots, j_y\} \cup \mathcal{I}_{\mathsf{eff}} < t$$

Since the only element that is different between the sets is j_w, these together imply that $j_w \notin \mathcal{I}_{\mathsf{eff}}$.

– Step 4d. Here, a party w' is added to $\mathcal{H}_i$ if $m'_{w'} = m^*$ but $m'_{w'+1} \neq m^*$. This combined with the fact that D is an admissible decoder, we get that,

$$|\{j_{w'}, \ldots, j_y\} \cup \{h_1, \ldots, h_{\hat{w}}\} \cup \mathcal{I}_{\mathsf{eff}}| \geq t$$

$$|\{j_{w'+1}, \ldots, j_y\} \cup \{h_1, \ldots, h_{\hat{w}}\} \cup \mathcal{I}_{\mathsf{eff}}| < t$$

Again, since these sets are different by just a single element, we have that $j_{w'} \notin \mathcal{I}_{\mathsf{eff}}$.

Lemma 8 also implies that, for any $i \in [z]$, if $\mathcal{T}_i$ had any party $y \in \mathcal{T}_i \cap \mathcal{I}_{\mathsf{eff}}$, then it cannot be in $\mathcal{H}_i$. Combined with Lemmas 7 and 13, we get that y must end up in $\mathcal{J}_i$, and hence in $\mathcal{I}^*$. Lastly, Lemma 7 implies that $\mathcal{I}^* \subseteq \mathcal{I}_{\mathsf{eff}}$. Combining the above, we get that $\mathcal{I}^* = \mathcal{I}_{\mathsf{eff}}$, i.e. the tracing algorithm does trace the decoder to the exact set of effective traitors. Lastly, since we know that the adversary outputs the effective set, we get that $\mathcal{I}^* = \mathcal{I}_{\mathsf{eff}} = \mathcal{I}$, meaning that the game will output 0, hence proving the theorem.

Handling probabilistic decoders. We now discuss how to generalize our tracing procedure to probabilistic, universal and admissible decoders, as defined in Definition 13. The Trace procedure now takes ϵ and $q(\lambda)$ as input, and we modify the procedure similar to how we handled probabilistic decoders for our confirmation algorithm, as described in Section 6.2. Informally, we replace checking whether $D(\{d_j\}_{j\in\mathcal{S}}) = m^*$ (for any set $\mathcal{S}$) with checking whether the estimated probability of D decrypting correctly, when given shares of $\mathcal{S}$, is more than $\epsilon - \frac{1}{2q(\lambda)}$. The rest of the logic remains the same.

We now formally describe all the changes to the procedure:

– (Step 4c) Run the decoder $W = \Omega(\lambda \cdot q(\lambda)^2)$ times on decryption shares of $\mathcal{J}_i$. Let $m_r \leftarrow\$ D(\{d_j\}_{j\in\mathcal{J}_i})$ for all $r \in [W]$. Set $\mathsf{ct} \leftarrow \Sigma_{r\in[W]} \mathbb{1}[m_r = m^*]$ and $\hat{p}_{\mathcal{J}_i} \leftarrow \frac{\mathsf{ct}}{W}$. If $\hat{p}_{\mathcal{J}_i} > \epsilon - \frac{1}{2q(\lambda)}$,
  - (Step 4(c)i) For each $r \in [y+1]$, run the decoder W times on decryption shares of $\{j_r, \ldots, j_y\}$. Let $m_{r,u} \leftarrow\$ D(\{d_v\}_{v\in\{j_r,\ldots,j_y\}})$ for all $u \in [W]$. Then, set $m_r \leftarrow m^*$ if $\frac{\Sigma_{u\in[W]}\mathbb{1}[m_{r,u}=m^*]}{W} > \epsilon - \frac{1}{2q(\lambda)}$, and $m_r \leftarrow \perp$ otherwise. The rest of the steps remain the same.

– (Step 4d) If $\hat{p}_{\mathcal{J}_i} < \epsilon - \frac{1}{2q(\lambda)}$, then,
  - (Step 4(d)ii) For each $r \in [u]$, run the decoder W times with decryption shares of the set $\mathcal{J}_i \cup \{h_1, \ldots, h_r\}$. Let $\hat{m}_{r,\ell} \leftarrow\$ D(\{d_v\}_{v\in\mathcal{J}_i\cup\{h_1,\ldots,h_r\}})$ for all $\ell \in [W]$. Then, set $\hat{m}_r \leftarrow m^*$ if $\frac{\Sigma_{\ell\in[W]}\mathbb{1}[m_{r,\ell}=m^*]}{W} > \epsilon - \frac{1}{2q(\lambda)}$, and $m_r \leftarrow \perp$ otherwise.
  - Step 4(d)iii remains the same, i.e. we find the smallest value $\hat{w}$ s.t. $\hat{m}_{\hat{w}} = m^*$.
  - In Step 4(d)iv, we run the decoder W times with decryption shares of the set $\{h_1, \ldots, h_{\hat{w}}\}$. Let $m_{h,u} \leftarrow\$ D(\{d_v\}_{v\in\{h_1,\ldots,h_{\hat{w}}\}})$ for all $u \in [W]$. If $\frac{\Sigma_{u\in[W]}\mathbb{1}[m_{h,u}=m^*]}{W} > \epsilon - \frac{1}{2q(\lambda)}$, then all parties in $\mathcal{J}_i$ are traitors, i.e. $\mathcal{I}^* \leftarrow \mathcal{I}^* \cup \mathcal{J}_i$.
  - (Step 4(d)v) Otherwise, for every $r \in [y+1]$, we run the decoder W times with decryption shares of the set $\{j_r, \ldots, j_y\} \cup \{h_1, \ldots, h_{\hat{w}}\}$, i.e. $m'_{r,u} \leftarrow\$ D(\{d_v\}_{v\in\{j_r,\ldots,j_y\}\cup\{h_1,\ldots,h_{\hat{w}}\}})$ for all $u \in [W]$. Next, set $m'_r \leftarrow m^*$ if $\frac{\Sigma_{u\in[W]}\mathbb{1}[m'_{r,u}=m^*]}{W} > \epsilon - \frac{1}{2q(\lambda)}$, and $m'_r \leftarrow \perp$ otherwise. The rest of the logic remains the same as Step 4(d)v.

We claim that the updated Trace algorithm satisfies tracing correctness. For any set $\mathcal{S}$, let $p_{\mathcal{S}}$ be the probability that the decoder decrypts correctly when given decryption shares of parties in $\mathcal{S}$. Then, running the decoder W times on shares of $\mathcal{S}$ gives us an estimate of this probability, $\hat{p}_{\mathcal{S}}$.

By Chernoff bound, we have that $|p_{\mathcal{S}} - \hat{p}_{\mathcal{S}}| < \frac{1}{4q(\lambda)}$ with high probability. Hence, $\hat{p}_{\mathcal{S}} > \epsilon - \frac{1}{2q(\lambda)}$ implies that $p_{\mathcal{S}}$ must be $\geq \epsilon$ with high probability. By adimissibility of the decoder D, this means that $|\mathcal{S} \cup \mathcal{I}|$ must be at least t, where $\mathcal{I}$ is the effective traitor set of D. The implication also holds in the other direction, i.e. $\hat{p}_{\mathcal{S}} < \epsilon - \frac{1}{2q(\lambda)}$ implies that $|\mathcal{S} \cup \mathcal{I}| < t$ with high probability. Hence, we can simply use the same proof technique as in the proof of Theorem 10 that the tracing algorithm does indeed trace the decoder to the set $\mathcal{I}$.

7 Conclusion and Future Directions

This work initiates the study of accountability for threshold decryption schemes, focusing on the notion of traitor tracing for such schemes. To the best of our knowledge, this is the first work to consider this problem, and it gives new definitions and constructions that satisfy them. We strongly believe that this may open an exciting avenue for research, as there are many natural open questions that arise from our work. We present two of them here, in the setting of tracing large traitor coalitions.

Thresholdizing additional traitor tracing schemes. As we explain in Section 3, a natural path to constructing a threshold decryption scheme with traitor tracing is by converting existing traitor tracing schemes into their threshold variant. In this work, we make the first step in this effort, showing how to convert the traitor tracing scheme of Boneh and Naor [14] to a threshold decryption scheme that allows for traitor tracing. In an upcoming follow-up work, we do the same for the recent traitor tracing scheme of Gong, Lou, and Wee [33]. An interesting open question is to adapt other traitor tracing schemes to the threshold setting. Natural candidates are other schemes based on private linear broadcast encryption (PLBE) from either pairing-based assumptions [20,32,55] or lattice-based assumptions [34,24], as well as the recent construction of Zhandry [56], which takes a different approach for implementing broadcast encryption with private revocation. Extending any of these works to the threshold setting will most likely require very different techniques than the ones we develop in this paper.

Detecting many traitors. Our definition of traitor tracing (Section 3) requires that the tracing algorithm Trace outputs at least *one* member of the traitor coalition. This is in line with previous definitions for traitor tracing in the non-threshold case. In the non-threshold setting, this is unavoidable: an adversary that corrupts $f > 1$ parties may very well still "use" just one of their secret keys when constructing the decoder. In threshold decryption, however, the decoder must use – in some intuitive sense – the secret keys of at least t parties in order to decrypt, or semantic security is broken. So one may consider a strengthening of our definition that requires that Trace outputs at least t corrupted parties. We observe that there is an inefficient scheme that does satisfy this definition, built from any semantically secure public key encryption $\mathcal{E}$. Consider the trivial threshold decryption scheme, in which each party i has its own secret-public key pair (sk_i, pk_i) and the overall public key is $pk = (pk_1, \ldots, pk_n)$. To encrypt a message m, we secret share m into shares $s_1, \ldots, s_n$ using a t-out-of-n secret sharing scheme, and encrypt the ith share under pk_i to obtain $c_i \xleftarrow{\$} \mathcal{E}.\mathsf{Enc}(pk_i, s_i)$. The final ciphertext is $c = (c_1, \ldots, c_n)$. It is not hard to see that this scheme has a tracing algorithm (à la PLBE) that can find at least t traitors given black-box access to a good decoder. The task is to satisfy this stronger requirement — tracing a decoder to t or more traitors — with an efficient construction. The combinatorial objects underlying most existing efficient traitor tracing constructions (PLBE and fingerprinting codes) are specifically tailored to catch just one traitor, and no more. So extending them to the task of catching t traitors seems to require new ideas, and is a very interesting open question.

Acknowledgments. This work was funded by NSF, DARPA, the Simons Foundation, UBRI, and NTT Research. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

1. A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari, and D. Yakira. Helix: A scalable and fair consensus algorithm resistant to ordering manipulation. Cryptology ePrint Archive, Report 2018/863, 2018. https://eprint.iacr.org/2018/863.
2. K. Babel, P. Daian, M. Kelkar, and A. Juels. Clockwork finance: Automated analysis of economic security in smart contracts. In *IEEE S&P*, pages 2499–2516. IEEE, 2023.
3. B. Bauer, G. Fuchsbauer, and J. Loss. A classification of computational assumptions in the algebraic group model. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151, Santa Barbara, CA, USA, Aug. 17–21, 2020. Springer, Heidelberg, Germany.
4. J. Bebel and D. Ojha. Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Archive, Report 2022/898, 2022. https://eprint.iacr.org/2022/898.
5. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press.
6. O. Billet and D. H. Phan. Efficient traitor tracing from collusion secure codes. In R. Safavi-Naini, editor, *ICITS 08*, volume 5155 of *LNCS*, pages 171–182, Calgary, Canada, Aug. 10–13, 2008. Springer, Heidelberg, Germany.
7. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46, Miami, FL, USA, Jan. 6–8, 2003. Springer, Heidelberg, Germany.
8. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.
9. D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In D. Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 226–243, San Jose, CA, USA, Feb. 13–17, 2006. Springer, Heidelberg, Germany.
10. D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464, Brisbane, Queensland, Australia, Dec. 2–6, 2018. Springer, Heidelberg, Germany.
11. D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353, Santa Barbara, CA, USA, Aug. 15–19, 1999. Springer, Heidelberg, Germany.
12. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.
13. D. Boneh, A. Kiayias, and H. W. Montgomery. Robust fingerprinting codes: A near optimal construction. In *Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management*, DRM '10, page 3–12, New York, NY, USA, 2010. Association for Computing Machinery.
14. D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 2008*, pages 501–510, Alexandria, Virginia, USA, Oct. 27–31, 2008. ACM Press.
15. D. Boneh, A. Partap, and L. Rotem. Accountable threshold signatures with proactive refresh. Cryptology ePrint Archive, Report 2022/1656, 2022. https://eprint.iacr.org/2022/1656.
16. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.
17. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In D. Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465, Santa Barbara, CA, USA, Aug. 27–31, 1995. Springer, Heidelberg, Germany.

18. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.
19. D. Boneh and V. Shoup. *A graduate course in applied cryptography (version 0.6)*. 2023. `cryptobook.us`.
20. D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 211–220, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press.
21. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499, Santa Barbara, CA, USA, Aug. 17–21, 2014. Springer, Heidelberg, Germany.
22. X. Boyen. The uber-assumption family: A unified complexity framework for bilinear groups. In *International Conference on Pairing-Based Cryptography*, pages 39–56. Springer, 2008.
23. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 90–106, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.
24. Y. Chen, V. Vaikuntanathan, B. Waters, H. Wee, and D. Wichs. Traitor-tracing from LWE made simple and attribute-based. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369, Panaji, India, Nov. 11–14, 2018. Springer, Heidelberg, Germany.
25. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270, Santa Barbara, CA, USA, Aug. 21–25, 1994. Springer, Heidelberg, Germany.
26. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.
27. P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy*, pages 910–927, San Francisco, CA, USA, May 18–21, 2020. IEEE Computer Society Press.
28. H. de Valence. The Penumbra protocol. link.
29. Y. Desmedt. Society and group oriented cryptography: A new concept. In C. Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 120–127, Santa Barbara, CA, USA, Aug. 16–20, 1988. Springer, Heidelberg, Germany.
30. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315, Santa Barbara, CA, USA, Aug. 20–24, 1990. Springer, Heidelberg, Germany.
31. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62, Santa Barbara, CA, USA, Aug. 19–23, 2018. Springer, Heidelberg, Germany.
32. S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 2010*, pages 121–130, Chicago, Illinois, USA, Oct. 4–8, 2010. ACM Press.
33. J. Gong, J. Luo, and H. Wee. Traitor tracing with $n^{1/3}$-size ciphertexts and $O(1)$-size keys from k-lin. In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 637–668, Lyon, France, Apr. 23–27, 2023. Springer, Heidelberg, Germany.
34. R. Goyal, V. Koppula, and B. Waters. Collusion resistant traitor tracing from learning with errors. In I. Diakonikolas, D. Kempe, and M. Henzinger, editors, *50th ACM STOC*, pages 660–670, Los Angeles, CA, USA, June 25–29, 2018. ACM Press.
35. A. Kiayias and M. Yung. Self protecting pirates and black-box traitor tracing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 63–79, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer, Heidelberg, Germany.
36. A. Kiayias and M. Yung. On crafty pirates and foxy tracers. In *Security and Privacy in Digital Rights Management*, pages 22–39. Springer, 2002.
37. A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 450–465, Amsterdam, The Netherlands, Apr. 28 – May 2, 2002. Springer, Heidelberg, Germany.
38. K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 145–157, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.
39. R. Li, Y. Li, Q. Wang, S. Duan, Q. Wang, and M. Ryan. Accountable decryption made formal and practical, 2023. available here.
40. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: Extended abstract. In M. K. Reiter and P. Samarati, editors, *ACM CCS 2001*, pages 245–254, Philadelphia, PA, USA, Nov. 5–8, 2001. ACM Press.

41. M. Naor and B. Pinkas. Threshold traitor tracing. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 502–517, Santa Barbara, CA, USA, Aug. 23–27, 1998. Springer, Heidelberg, Germany.
42. J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, Aug. 16–20, 2021. Springer, Heidelberg, Germany.
43. K. Nuida. A general conversion method of fingerprint codes to (more) robust fingerprint codes against bit erasure. In K. Kurosawa, editor, *ICITS 09*, volume 5973 of *LNCS*, pages 194–212, Shizuoka, Japan, Dec. 3–6, 2010. Springer, Heidelberg, Germany.
44. Osmosis. link.
45. D. H. Phan. Traitor tracing for stateful pirate decoders with constant ciphertext rate. In *Progressin of VI-ETCRYPT 2006*, volume 4341 of *LNCS*, pages 354–365. Springer, 2006.
46. K. Qin, L. Zhou, and A. Gervais. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy*, pages 198–214, San Francisco, CA, USA, May 22–26, 2022. IEEE Computer Society Press.
47. A. Rondelet and Q. Kilbourn. Threshold encrypted mempools: Limitations and considerations, 2023. available here.
48. L. Rotem. Revisiting the Uber Assumption in the Algebraic Group Model: Fine-Grained Bounds in Hidden-Order Groups and Improved Reductions in Bilinear Groups. In D. Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography (ITC 2022)*, volume 230, pages 13:1–13:13, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
49. L. Rotem and G. Segev. Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 366–389, Durham, NC, USA, Nov. 16–19, 2020. Springer, Heidelberg, Germany.
50. K. Sako. An auction protocol which hides bids of losers. In H. Imai and Y. Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 422–432, Melbourne, Victoria, Australia, Jan. 18–20, 2000. Springer, Heidelberg, Germany.
51. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.
52. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 1–16, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.
53. T. Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. Cryptology ePrint Archive, Report 2006/383, 2006. `https://eprint.iacr.org/2006/383`.
54. G. Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2), may 2008.
55. H. Wee. Functional encryption for quadratic functions from k-lin, revisited. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 210–228, Durham, NC, USA, Nov. 16–19, 2020. Springer, Heidelberg, Germany.
56. M. Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 652–682, Santa Barbara, CA, USA, Aug. 17–21, 2020. Springer, Heidelberg, Germany.

A A BT-KEM Construction from any Threshold KEM

We briefly describe how one can generically construct a BT-KEM scheme any threshold KEM scheme. Let $\mathcal{E}$ be such a threshold KEM scheme. A naive first attempt of constructing a BT-KEM from $\mathcal{E}$ is already implicit in our discussion at the beginning of Section 4. Casted as a BT-KEM scheme, this naive construction uses 2ℓ t-out-of-n copies of $\mathcal{E}$, a left one and a right one for each position $j \in [\ell]$. The public key of our BT-KEM is now composed of ℓ pairs of $\mathcal{E}$ public keys $(pk_0^{(1)}, pk_1^{(1)}), \ldots, (pk_0^{(\ell)}, pk_1^{(\ell)})$. Similarly, the secret key of party $i \in [n]$ consists of 2ℓ secret keys, one for each copy of $\mathcal{E}$. To encapsulate a key to position $j \in [\ell]$, the encapsulation algorithm Enc invokes $\mathcal{E}.\mathsf{Enc}$ twice: once for each of the copies of $\mathcal{E}$ at position j. It thus obtains two keys k_0 and k_1 and two ciphertexts c_0 and c_1. It samples a key k from the key space of $\mathcal{E}$, and one-time pads it with both k_0 and k_1 to obtain ciphertexts c'_0 and c'_1. It then outputs they key k and the ciphertext

$c = ((c_0, c'_0), (c_1, c'_1))$. As we discussed in Section 4, this construction does not provide two sides correctness.

To fix this issue, we add more information to the ciphertext. In each position $j \in [\ell]$, we initialize $2(t-1)$ additional copies of $\mathcal{E}$: for $s = 1, \ldots, t-1$, we sample a set of left keys for an s-out-of-n instance of $\mathcal{E}$ and a set of right keys for a $(t-s)$-out-of-n instance of $\mathcal{E}$. This is in addition to the two sets of t-out-of-n keys. All in all, we now have t sets of left keys and t sets of right keys at each position ($2t\ell$ sets of keys overall). When encapsulating to position $j \in [\ell]$, the encapsulation algorithm now repeats the following for $s = 1, \ldots, t-1$: it 2-out-of-2 secret shares k to k'_0 and k'_1. It encrypts k'_0 under the s-out-of-n left key at position j and k'_1 under the $(t-s)$-out-of-n right key. Now, if a coalition has s left keys for some $s \in \{0, \ldots, t\}$ and $t-s$ right keys, it can decapusalte by using its left keys to recover k'_0 and its right keys to recover k'_1, and reconstructing the encapuslated key k.

The construction that we just sketched demonstrates that BT-KEM can be constructed from the minimal building block which is a standard threshold KEM scheme. However, due to its long keys and ciphertexts, it is mainly of foundational interest. In Section 5 we describe much more efficient constructions from specific assumptions.

B A Lower Bound for Tracing of General Decoders

We now prove that if we pose no restrictions on the decoder outputted by a small coalition, then tracing is impossible if the underlying threshold decryption scheme is robust (recall Defintion 2). Let $\mathcal{E}$ be any robust threshold, let $n \in \mathbb{N}$, $t < n$, and $f < t$, let $(pk, pkc, sk_1, \ldots, sk_n, vk, tk) \xleftarrow{\$} \mathcal{E}.\mathsf{KeyGen}(1^\lambda, n, t)$, and let m be an arbitrary message in the message space of $\mathcal{E}$, and let $c \xleftarrow{\$} \mathcal{E}.\mathsf{Enc}(pk, m)$. Consider the following distribution $\mathcal{D} = \mathcal{D}(n, t, f, pk, pkc, sk_1, \ldots, sk_n, vk)$ over decoders:

1. Sample a uniformly random subset $\mathcal{I} \subseteq [n]$ of size f.
2. For each $i \in \mathcal{I}$, compute $d_i \leftarrow \mathcal{E}.\mathsf{Dec}(sk_i)$
3. Construct a decoder D, that takes in an arbitrary number of decryption shares $\{d_j\}_{j \in \mathcal{J}}$ for some subset $\mathcal{J} \subseteq [n]$. Since $\mathcal{E}$ is robust, the subset $\mathcal{J}$ is recoverable from the shares $\{d_j\}_{j \in \mathcal{J}}$. D is defined as follows:
   (a) If $|\mathcal{J}| \neq t - f$, then output $\perp$.
   (b) If $\mathcal{I} \cap \mathcal{J} \neq \emptyset$ then output $\perp$.
   (c) If not aborted, compute $m \leftarrow \mathcal{E}.\mathsf{Combine}(pkc, c, \mathcal{J} \cup \mathcal{I}, \{d_i\}_{i \in \mathcal{J} \cup \mathcal{I}})$.
4. Output the decoder D.

Note that the decoder outputted by the distribution depends only on the secret keys corresponding to the subset $\mathcal{I}$. Hence, it can be computed by a small traitor coalition possessing $f < t$ keys. The decoder is also useful, in the sense that for any subset $\mathcal{J} \subset [n]$ of size $t - f$ that is disjoint from $\mathcal{I}$, feeding the decoder $\{d_j \leftarrow \mathcal{E}.\mathsf{Dec}(sk_j, c)\}_{j \in \mathcal{J}}$ will make it output the encrypted message m.

We claim that given oracle access to a decoder D sampled from this distribution, it is very unlikely to make it output anything but $\perp$.

Theorem 11. *Let $n, t, f, pk, pkc, sk_1, \ldots, sk_n, vk, tk$ and $\mathcal{D}$ be defined as above. Then, for any probabilistic polynomial-time algorithm $\mathcal{B}$ making at most $Q = Q(\lambda)$ queries to its oracle, it holds that*

$$\Pr\left[\mathcal{A} \neq \{\perp\}\right] \leq \frac{Q}{\binom{n}{f} - Q \cdot \binom{n-(t-f)}{f}},$$

where $\mathcal{A}$ is the set of all oracle answers in a random execution of $\mathcal{B}^D(n, t, f, pk, pkc, sk_1, \ldots, sk_n, vk, tk)$, and the probability is over $D \stackrel{\$}{\leftarrow} \mathcal{D}$ and the random coins of $\mathcal{B}$.

Before proving the theorem, we wish to better understand its implications. To this end, consider the case $t = n/3$ and $f = t/2$. Note that we can rewrite the bound from Theorem 11 as

$$\frac{Q}{\binom{n}{f} - Q \cdot \binom{n-(t-f)}{f}} = \frac{Q}{\binom{n}{f} \cdot \left(1 - Q \cdot \binom{n-(t-f)}{f} / \binom{n}{f}\right)}.$$

We now turn to bound the term $\binom{n-(t-f)}{f} / \binom{n}{f}$. By Stirling approximation, we obtain that

$$\begin{aligned}
\frac{\binom{n-(t-f)}{f}}{\binom{n}{f}} &= \frac{(n-(t-f))!(n-f)!}{n!(n-t)!} \\
&\leq e^2 \sqrt{\frac{(n-t+f)(n-f)}{n(n-t)}} \cdot \frac{\left(\frac{n-t+f}{e}\right)^{n-t+f} \cdot \left(\frac{n-f}{e}\right)^{n-f}}{\left(\frac{n}{e}\right)^n \cdot \left(\frac{n-t}{e}\right)^{n-t}} \\
&\leq e^2 \sqrt{n} \cdot \frac{(n-t+f)^{n-t+f} \cdot (n-f)^{n-f}}{n^n \cdot (n-t)^{n-t}}
\end{aligned} \tag{16}$$

Plugging in $t = n/3$ and $f = t/2$, we get that the term from Eq. (16) is upper bounded by $e^2\sqrt{n} \cdot 0.97^n$. In particular, assuming that $Q < 0.97^{-n}/(2e^2\sqrt{n})$ (and in particular for any polynomial Q), it holds that

$$\begin{aligned}
\Pr\left[\mathcal{A} \neq \{\perp\}\right] &\leq \frac{Q}{\binom{n}{f} - Q \cdot \binom{n-(t-f)}{f}} \\
&\leq \frac{Q}{\binom{n}{f} \cdot (1 - Qe^2\sqrt{n} \cdot 0.97^n))} \\
&< \frac{Q}{2\binom{n}{f}}
\end{aligned}$$

Invoking Stirling's approximation one more time, we can show that

$$\binom{n}{f} \geq \frac{6}{e\sqrt{10\pi n}} \cdot \left(\frac{1}{\left(\frac{1}{6}\right)^{1/6} \cdot \left(\frac{5}{6}\right)^{5/6}}\right)^n > \frac{6}{e\sqrt{10\pi n}} \cdot (1.569)^n$$

Putting everything together, we obtain that

$$\Pr\left[\mathcal{A} \neq \{\perp\}\right] \leq Q \cdot \frac{e\sqrt{10\pi n}}{12 \cdot (1.569)^n} = Q \cdot 2^{-\Omega(n)}$$

We now turn to prove Theorem 11.

Proof. Assume without loss of generality that $\mathcal{B}$ only queries D on subsets of size $t-f$. Since $\mathcal{E}$ is robust, each decryption share is deterministically associated with some party $j \in [n]$. To make D outputs anything other than $\perp$, $\mathcal{B}$ has to query D on decryption shares from a subset $\mathcal{J}$ of parties that is of size $t-f$ and is disjoint from $\mathcal{I}$. If this does not happen, then D always outputs $\perp$ regardless of the decryption shares themselves given to it. Hence, for the remainder of the proof, we will ignore the decryption shares and just think of it as if $\mathcal{B}$ queries D on subsets $\mathcal{J}_1, \ldots, \mathcal{J}_Q$. We will prove an upper bound on the probability that there is a $k \in [Q]$ for which $\mathcal{J}_k \cap \mathcal{I} = \emptyset$.

Let $\mathcal{S}_f$ denote the collection of all f-size subsets of $[n]$. For $i = 1, \ldots, Q-1$, denote

$$\mathcal{S}_{-i} = \{\mathcal{T} \in \mathcal{S}_f \ : \ \mathcal{T} \cap \mathcal{J}_i = \emptyset\}.$$

That is, $\mathcal{S}_{-i}$ is defined to be the collection of all f-size subsets of $[n]$ that do not intersect $\mathcal{J}_i$. Let $\mathcal{W}_0 = \mathcal{S}_f$ and for $i = 1, \ldots, Q-1$ let $\mathcal{W}_i = \mathcal{W}_{i-1} \setminus \mathcal{S}_{-i}$. In words, the set $\mathcal{W}_i$ is the collection of all f-size subsets of $[n]$, when we remove from it all subsets that do not intersect at least one of the subsets $\mathcal{J}_1, \ldots, \mathcal{J}_i$.

Moreover, for $i \in [Q]$, the size of $\mathcal{W}_{i-1}$ is at least

$$|\mathcal{W}_{i-1}| \geq |\mathcal{S}_f| - (i-1) \cdot \binom{n-(t-f)}{f} \tag{17}$$

This can be seen by induction on i. For $i = 1$, Eq. (17) holds by definition. For $i > 1$, we have that

$$\begin{aligned} |\mathcal{W}_{i-1}| &= \left|\mathcal{W}_{i-2} \setminus \mathcal{S}_{-(i-1)}\right| \\ &\geq |\mathcal{W}_{i-2}| - \left|\mathcal{S}_{-(i-1)}\right| \\ &\geq |\mathcal{S}_f| - (i-2) \cdot \binom{n-(t-f)}{f} - \left|\mathcal{S}_{-(i-1)}\right| \qquad (18) \\ &= |\mathcal{S}_f| - (i-1) \cdot \binom{n-(t-f)}{f}. \qquad (19) \end{aligned}$$

Eq. (18) is the induction hypothesis, and Eq. 19 holds since the number of f-size subsets that do not intersect $\mathcal{J}_{i-1}$ is $\binom{n-(t-f)}{f}$.

For $i = 1, \ldots, Q$ let Hit_i denote the event in which one of the first i queries $\mathcal{B}$ made to D was answered with something other than $\perp$. For $i = 1$ we let Hit_0 denote empty event; that is, $\Pr[\mathsf{Hit}_0] = 0$. The main observation is this: let $i \in [Q]$ and suppose that $\overline{\mathsf{Hit}_{i-1}}$ occurred. Then, conditioned on the view of $\mathcal{B}$ after $i-1$ queries, the subset $\mathcal{I}$ is uniformly distributed in the set

$\mathcal{W}_{i-1}$. Hence,

$$
\begin{aligned}
\Pr\left[\mathcal{A} \neq \{\perp\}\right] &= \bigcup_{i=1}^{Q} \Pr\left[\mathsf{Hit}_i \;\middle|\; \overline{\mathsf{Hit}_{i-1}}\;\right] \\
&\leq \sum_{i=1}^{Q} \Pr\left[\mathsf{Hit}_i \;\middle|\; \overline{\mathsf{Hit}_{i-1}}\;\right] \\
&= \sum_{i=1}^{Q} \frac{1}{|\mathcal{W}_{i-1}|} \\
&< \frac{Q}{|\mathcal{S}_f| - Q \cdot \binom{n-(t-f)}{f}} \\
&= \frac{Q}{\binom{n}{f} - Q \cdot \binom{n-(t-f)}{f}}.
\end{aligned}
$$

This concludes the proof of the Theorem.