CCA-Secure Traceable Threshold (ID-based) Encryption and Application

Rishiraj Bhattacharyya
University of Birmingham, UK
rishiraj.bhattacharyya@gmail.com

Jan Bormet
TU Darmstadt, Germany
jan.bormet@tu-darmstadt.de

Sebastian Faust
TU Darmstadt, Germany
sebastian.faust@tu-darmstadt.de

Pratyay Mukherjee
Supra Research, India
pratyay85@gmail.com

Hussien Othman
TU Darmstadt, Germany
hussien.othman@gmail.com

Abstract

A recent work by Boneh, Partap, and Rotem [Crypto'24] introduced the concept of traceable threshold encryption, in that if t or more parties collude to construct a decryption box, which performs decryptions, then at least one party's identity can be traced by making a few black-box queries to the box. This has important applications, e.g., in blockchain mempool privacy, where collusion yields high financial gain through MEVs without any consequence – the possibility of tracing discourages collusion. Nevertheless, their definitions leave room for exploitation as they only achieve CPA security and do not consider inconsistency in decryption via different participating sets.

This paper proposes stronger definitions of traceable threshold encryption, which supports CCA-security and consistency. Our main approach considers identity-based variants of traceable encryption (which we also define). It converts that to a CCA-secure construction, adapting two generic transformations, first using a one-time signature and then a fingerprinting code. We put forward two efficient instantiations of our identity-based scheme with different merits: our first construction is based on Boneh-Franklin IBE [Crypto'01] and has constant size ciphertexts but quadratic size public keys – this is proven secure based on XDH and BDDH. Our second construction is based on Boneh-Boyen IBE [Eurocrypt'04]. It supports both constant-size ciphertexts and constant-size public keys – this is proven secure based on a variant of the uber assumption over bilinear pairings. Our concrete analysis shows that the first construction's ciphertext is much (~6x) smaller than the second construction. Finally, we extend the definitions to support consistency and achieve it by adjoining an efficient, non-interactive proof of correct encryption.

1 Introduction

Threshold cryptographic divides a cryptographic procedure into multiple subprotocols, distributing them among several (say n) participants. This ensures that no single participant holds the entire secret, avoiding a single point of failure. Only a subset of participants, meeting a predefined "threshold" (say, $t \leq n$) can collaboratively perform the cryptographic operation or reconstruct the secret. Threshold cryptography traditionally guarantees security when at most $(t-1)$ parties are corrupt (and possibly colluding). Specifically, if more than t parties are corrupted by the adversary, then all bets are off, as the adversary gets to know the entire secret! However, in many practical applications of threshold cryptography if t or more parties collude, then the colluders can exploit the system for significant financial gain. Without a mechanism for discouraging or even detecting such behavior, this possibility may turn out to be rationally viable as well.

For example, let us consider the problem of mempool privacy. In many blockchain ecosystems, such as Ethereum, front-running attacks, and variants [33, 52, 53], on mempools[1] are commonplace. Here, the validators of a blockchain leverage their advantage of observing transactions before they are finalized. They selectively finalize specific blocks or put transactions from the mempool in a specific order and, depending on that, may extract substantial financial gains (aka MEVs [29]). Existing solutions [2, 19, 25, 26, 48] broadly incorporate threshold decryption techniques, in which transactions in the mempool are encrypted, and the decryption keys are distributed among the validators in a t out of n threshold access structure. Transactions are decrypted only when at least t validators agree to decrypt, which takes place only after a certain time when no further MEVs can be derived. However, in practice, without additional mechanisms, there is no way to enforce $< t$ corruption. In particular, if $\geq t$ validators continue to extract MEVs by just decrypting transactions earlier than scheduled, no one can even detect that. In fact, it is clearly a rational choice for the validators to derive MEVs via collusion.

Towards mitigating this kind of problem in the context of threshold encryption, the work of Boneh, Partap, and Rotem [15] (henceforth, BPR24) recently put forward the notion of *traceable threshold encryption*, which allows a designated party, called the *tracer* (who may possess a secret tracing key) to catch at least one traitor from the collusion of at least t parties in the context of threshold decryption. More precisely, if the colluding group (of $\geq t$ parties) constructs a decryption box that, on a given ciphertext, returns the corresponding decrypted value, then with a few queries to the box, the tracer can output the identity of at least one of the colluders. Therefore, with an adequate penalty in place (for example, via slashing), validators are much less likely to collude, as the financial benefit of colluding over not colluding is not clear anymore. In the same paper [15], the authors proposed two traceable threshold encryption schemes with different merits, both based on finger-printing codes [18].

While the constructions in BPR24 [15] are quite efficient, their security notions only provide weaker security guarantee – they achieve only CPA-security, while CCA-security is considered the "gold standard" for encryptions [5]. In this work, we address this

[1] Roughly, validators locally maintain a mempool that contains all transactions sent to a blockchain that have not been finalized in a block.

natural problem by extending the notion of traceable threshold encryption to CCA-security. Moreover, from a practical perspective in the context of mempool privacy, two concrete practical attacks are not captured by BPR24: (i) Since their definition only considers CPA security, a non-malleability attack is easy to execute, for example, by blindly mauling an encrypted transaction to something related. This related transaction may be decrypted earlier, thereby violating transaction privacy. We stress that such an attack also works when the plaintexts are signed, which is usually the case for mempool data. This is because, even if blind mauling makes the signature invalid, the plaintext would still get exposed. Hence, a malicious validator can frontrun the original transaction in the next block. Instead, CCA security would guarantee that no information is leaked when the ciphertext is mauled. (ii) The notion does not guarantee a *consistency* of decryption, which ensures that the decryption results into the same plaintext as long as *any* set of t decryptors participate, regardless of the specific members of the set – this can be exploited, for example, by adversarially crafting a ciphertext which results into different decryptions based on the set of decryptors. In fact, these attacks are not only outside their definitions but they can be easily executed on their constructions, simply exploiting the homomorphic structures (see Appendix A for more details). This begets not only an extension of their notions but also new, stronger constructions secure therein.

Our Contribution. In this paper, we primarily focus on addressing these issues comprehensively first by extending BPR24's notion to *CCA-security* and then adding *consistency*; finally, we propose two new efficient constructions satisfying our notions. In particular, our contributions are as follows:

- **New Stronger Definitions.** We formalize the notion of *CCA-security* in the setting of traceable threshold key encapsulation mechanisms (TT−KEM).[2] En route, we also define the notion of identity-based traceable KEM, which supports a threshold key-derivation – we use this as a building block towards CCA security. Finally, we augment our CCA-security definitions to support the crucial *consistency* property. Our new definitions capture the above.
- **Constructing Stronger** TT−KEM **Schemes.** We construct CCA-secure TT−KEM schemes generically from a CCA-secure bipartite threshold KEM scheme (BT−KEM) – a CPA-secure version of this primitive was defined in BPR24, we augment that to CCA-security. Our generic transformation is an adaptation from BPR24's techniques in the context of CPA-security and also relies on the fingerprinting codes [17]. However, as we explain later in Section 1.1, the CPA-secure construction from BPR24 is not compatible with the existing CPA-to-CCA transformation. So we resort to a new approach via IBE. We construct CCA-secure BT−KEM generically from identity-based BT−KEM (henceforth BTIB−KEM, which we define here adapting the PKE definition from BPR24) lifting the one-time-signature based technique due to [11]. Finally, we instantiate BTIB−KEM with two concrete constructions:
  - (i) Our first construction is based on the Boneh-Franklin IBE [12] and relies on XDH and BDDH assumptions on bilinear pairing groups. Instantiating with this construction, we obtain a CCA-TT−KEM, which has a constant size ciphertext but a larger public key, which scales quadratically with the number of parties.
  - (ii) Our second construction is based on the Boneh-Boyen IBE [7]. Instantiating with this, we obtain a TT−KEM scheme, where both ciphertext and public key are of constant size. However, the ciphertext is concretely larger than the first construction. The security proof becomes much more involved and relies on a variant of Uber assumptions [8, 20], which hold in generic groups. This construction additionally supports hierarchical IBE, and using a generic transformation using fingerprinting codes, similar to the aforementioned BT−KEM to TT−KEM transformation, we can obtain a CCA-secure traceable threshold IBE (TTIB−KEM) scheme – this may be of independent interest.[3]
- **Achieving Consistency.** Finally, we extend our CCA-secure TT−KEM constructions to achieve the consistency property by using an efficient non-interactive zero-knowledge (NIZK) proof of the well-formedness of the ciphertext. However, the challenging part is to resolve the apparently contradictory requirements of public verifiability of ciphertext well-formedness and traceability, which relies on the colluding party's inability to detect a malformed ciphertext. We resolve this by including a simulation trapdoor (for NIZK) into the tracing key which enables only the tracer to produce "false proofs". We also note that a similar strategy can not be incorporated to achieve CCA-security, because that should hold also against a malicious tracer. We discuss this in more detail in the next section (Section 1.1).

1.1 Technical Overview

In this section, we provide an overview of our techniques. We start with the setting, which is the same as BPR24.

The Setting. We consider an arbitrary threshold access structure for decryption in that any t out of n parties together can decrypt any ciphertext and reconstruct the whole decryption key. Among them, any set of parties (possibly $\geq t$) can collude to construct a decryption/decoder box, called D.[4] BPR24 allows for decoders that can be used to learn *any* non-trivial information about the encrypted message (not even the full message). This is formally modeled through the distinguishing advantage. To allow for tracability, the threshold encryption scheme is equipped with a tracing algorithm, which would find out the identity of at least one colluder while making a limited number of black-box queries to D, possibly using a private tracing key. However, note that as long as the corruption is within the threshold t, the privacy (CCA-security in our

[2] Similar to BPR24 [15] we work with the notion of KEM for cleaner constructions and presentations, all our constructions can be converted to usual encryption schemes following the standard notion of hybrid encryption (KEM-DEM paradigm).

[3] We stress that, for the transformation from BTIB−KEM to CCA-BT−KEM only CPA security of the former suffices.

[4] This captures the practical scenario, in that the colluding parties may collaborate to produce and sell a device containing everyone's shares. The device can be purchased, possibly in a black market, and then used by anyone to decrypt any ciphertext.

case) should hold even against the tracer, who owns the tracing key. In the CCA-security game, the adversary is given the tracing key to model this. Nonetheless, once t or more parties collude to construct a decryption box, no privacy can be guaranteed. Instead, in that case, the tracer is responsible for finding a colluder by using the decoder box as an oracle. In reality, this captures the scenario when the tracer (e.g., law enforcement) buys a copy of the decoder box and runs it with various inputs at their own disposal.

Overview of BPR24 [15]. Boneh and Naor [14] constructed a public-key encryption scheme with traitor-tracing using fingerprinting codes [17]. Intuitively, fingerprinting codes enable one to trace information about the original codewords from noisy ones – this helps in tracing when secret keys are encoded with such codes. The constructions of BPR24 used fingerprinting codes in a similar fashion, with adequate changes to make it work in a threshold setting. Technically, the main idea is to first construct a building block (defined in BPR24) called bipartite threshold KEM (BT−KEM) and then convert it to a TT−KEM using a generic transform. Informally, in BT−KEM, a secret key consists of 2ℓ elements for some parameter ℓ. For each index $i \in [\ell]$, there are two keys, a left key and a right key. Each ciphertext also contains a left and a right part and is associated with an index $j \in [\ell]$. The correctness is *two-sided*, in that any collection of keys from t parties possibly holding a mix of left or right keys for index j, are sufficient to decrypt any ciphertext; security is *one-sided*, in that a set (possibly of size $\geq t$) of parties, all of them holding only the left key (resp. right key) for index j can not distinguish the corresponding right part (resp. left part) of the ciphertext from a random value. In particular, this one side-security enables the tracing functionality similar to [14].

Specifically, BPR24's generic transformation from BT−KEM to TT−KEM works as follows: The setup phase samples a fingerprinting code of n words of size ℓ. It runs the key-generation of BT−KEM with parameter ℓ to generate 2ℓ secret keys, a pair for each index $j \in [\ell]$. Each party-i receives a unique secret key share which consists of exactly ℓ elements chosen among the 2ℓ pairs according to the i-th finger-printing codeword, such that a party obtains exactly one key (either left or right) for each index $i \in [\ell]$. To encrypt a message m first a random index $j \in [\ell]$ is chosen, and then it is encrypted with respect to the j-th index to generate the left and right ciphertext pair (c_0, c_1). Decryption works with any subset of shares of left or right keys. Traceability is achieved using one-sided security plus the tracing mechanism of the finger-printing codes: carefully crafted combinations of invalid and valid ciphertexts are queried to D, and based on whether invalidity is detected or not, the tracer can recover (partial) information about the codewords corresponding to the colluders, thereby narrowing down the plausible set of the colluders, and eventually a colluder's identity. The intuition is, if, e.g., all the colluders have only left shares for index j, then, by the one-sided security of BT−KEM, the decoder D cannot distinguish valid c_1 from an invalid one, and hence would return a valid decryption. So the tracer can conclude that all colluders correspond to j-th index being 0 and so on.

Failed Attempts for CCA. Now, the most natural way to try to achieve CCA-security from the state of the art (namely BPR24) is to apply existing CPA-to-CCA transformation techniques to the BPR24's constructions. However, the main challenge is to ensure

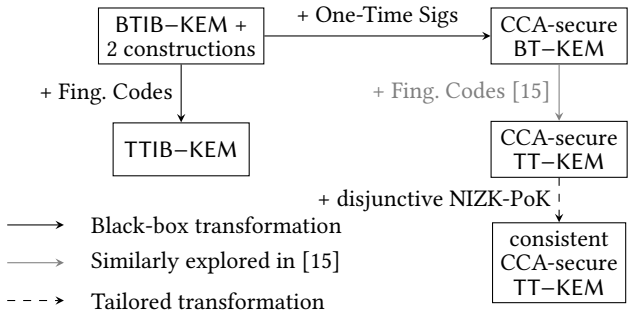

Figure 1: Overview of the techniques used in this paper.

that CCA-security, which guarantees non-malleability, does not interfere with traceability, which does require some sort of malleability, as explained above. The existing CPA-to-CCA approaches fail to overcome precisely this issue. To illustrate let us first consider the Fujiaski-Okamoto [34, 35] transformation: the encapsulation mechanism encrypts a random string r via a CPA-secure encryption (where the encryption randomness is deterministically derived from r) to produce the ciphertext c. Then the random message (or key) is derived by hashing r. That is, $c = \mathsf{Encrypt}(pk, r)$ and $k = \mathcal{H}(r)$. The main argument in proving CCA security is, as the adversary never gets to decrypt the challenge ciphertext c, it is unable to find the value of r, even with the decryption of some other $c' = \mathsf{Encrypt}(pk, r')$, and thus k appears completely random assuming the hash to be a random oracle. Let us now try to apply this to the above CPA TT−KEM: encrypt the random string r using BT−KEM to get a ciphertext pair (c_0, c_1) with respect to some index j. Any of c_0 or c_1 can be decrypted to derive the random string r and hence the correct key. Given a decryption oracle, one can query a mauled ciphertext (c_0, c_1^*). Since the BT−KEM decryption decrypts one of the ciphertexts, it may just decrypt the left one and return r, breaking security. The crux of the problem here is, decryption in the BPR24 construction does not have a mechanism to publicly verify that the pair (c_0, c_1) is indeed a valid ciphertext.[5]

An alternative is to consider the Naor-Yung [44] paradigm (or its variants, such as Cramer-Shoup [28] using hash-proof), which essentially addresses this public verifiability problem by attaching a non-interactive zero-knowledge proof of the fact that two ciphertexts encrypt the same message. Unfortunately, this approach does not work because the decoder box can just detect if an invalid ciphertext is queried, yielding $\perp$. Hence, one-sided security can not be guaranteed, and it is unclear how to make the tracing algorithm work.[6]

Our Approach to achieve CCA-security. The main technical challenge here is to ensure that CCA-security, which guarantees non-malleability, does not interfere with traceability, which does require some sort of malleability, as explained above. We resolve this issue

[5]We note that, since the BT−KEM decryption uses one of the left or right keys, but not both, it can not check equality after decryption.

[6]One might attempt to resolve this issue by including a simulation trapdoor into the private tracing key – this would enable the tracer to produce simulated proofs of false statements. Nonetheless, this would not work because the CCA-adversary gets access to the tracing key, thus is able to produce false proofs and break CCA security.

using a different route. In particular, we first extend the notion of BT−KEM to support identity – we call this new notion the bipartite threshold identity-based KEM (BTIB−KEM). Then we use a variant of IBE-to-CCA generic transformation by Boneh et al. [11] to construct a CCA-secure BT−KEM. Finally, adapting the generic transformation of BPR24 to the CCA setting, we convert CCA-secure BT−KEM to CCA secure TT−KEM using fingerprinting like the above.

From BTIB−KEM *to CCA*-BT−KEM. Let us start by recalling the idea of Boneh et al. [11] which uses a one-time signature to generically transform an IBE to a CCA-secure PKE scheme. Let the master key pair of the IBE scheme be $(\mathsf{mpk}, \mathsf{msk})$. Then the encryption algorithm of the PKE works as follows: sample a fresh signing-verification key pair $(\mathsf{sk}, \mathsf{vk})$, and use vk as the identity to encrypt the message m to ciphertext c. Then sign c with sk to compute signature σ. The final ciphertext is (vk, c, σ). Decryption works by first checking the signature, then deriving the identity key using the master key, and decrypting c. If the signature fails to verify, it outputs $\perp$. Intuitively, CCA-security follows because (i) if the adversary makes decryption queries with a ciphertext which has $\mathsf{vk}' \neq \mathsf{vk}$, then the ciphertext is unrelated to the challenge ciphertext; (ii) if the adversary makes a decryption query with a ciphertext which has $\mathsf{vk}' = \mathsf{vk}$, the ciphertext is either invalid or can serve as a forgery to the signature scheme. Now we apply the same technique to BTIB−KEM in a way such that one is able to generate a correct signature for a malformed ciphertext such that traceability is possible. To see this, we note that in our modified one-sided security game for CCA secure BT−KEM, the challenger generates two ciphertexts $c^{(0)} = (c_0^{(0)}, c_1^{(0)})$ and $c^{(1)} = (c_0^{(1)}, c_1^{(1)})$. Then, it challenges the adversary with a possibly malformed ciphertext c^*, such as a mix of $c^{(0)}$ and $c^{(1)}$ (i.e., takes one side from each ciphertext). Here, the adversary can distinguish between the cases when the ciphertext is valid and when it is malformed by verifying the signature. This is fixed by modifying the one-sided security definition, in that the two ciphertexts $c^{(0)}$ and $c^{(1)}$ are encrypted to the same identity key vk (as tags) and then sign the possibly malformed c^* using sk. Finally, we show that our modified one-sided security game does not break tracing by discussing the required changes needed to the tracing algorithm of [15].

To conclude, we argue in Section 3.3 that the bipartite-to-traceable transformation of [15] and this work both preserve CCA-security. Hence, plugging any CCA-secure BT−KEM in the black-box transformation in [15] results in a CCA-secure TT−KEM.

Efficient BTIB−KEM. Finally we present two instantiations of BTIB−KEM. We achieve our first construction based on Boneh-Franklin IBE scheme [12] – this can be thought of as an identity-based extension of BPR24. This construction enjoys a short ciphertext of only 2 group elements.However, the construction requires a quadratic-size public key. Our second construction enjoys a constant-size public key. We achieve this by using Boneh-Boyen two-level (hierarchical) IBE [7]. The construction is more involved. However, in concrete terms, this construction has larger ciphertext (12 group elements compared to 2 in the first), though it offers better asymptotic parameters. In practical settings, such as mempool privacy, typically short ciphertexts are more desired. Nevertheless, the second construction supports a hierarchical feature. Note that from a two-level hierarchical BTIB−KEM, we can achieve a CCA-secure traceable IBE using the BPR24 transformation. This may be of independent interest as identity-based encryption schemes are useful in popular mempool privacy solutions as well [48].

A summary of our techniques and results is given in Figure 1.

1.2 Further Related Works

We only briefly discuss related works that aim at preventing collusion in the threshold setting. For a more comprehensive discussion, we refer the reader to Appendix B. To the best of our knowledge, there are no other threshold traceable decryption constructions besides [15]. Some other recent works study collusion resistance of secret sharing [16, 31, 38], but it is not clear how to extend these techniques for constructing threshold crypto schemes.

1.3 Roadmap

In Section 2 we present some preliminaries. Then, in Section 3 we discuss the definition of CCA-secure TT−KEM and present a transformation from CCA-secure BT−KEM to CCA-secure TT−KEM. In Section 4, we introduce identity-based BT−KEM (BTIB−KEM) as our main building block and show how to construct CCA-secure BT−KEM from BTIB−KEM. We present two BTIB−KEM constructions in Section 5. In Section 6, we discuss the efficiency of our constructions. Finally, we discuss consistency in Section 7.

2 Preliminaries

Notation. We denote the security parameter as $\lambda \in \mathbb{N}$, the key space for KEM protocols as $\mathcal{K}(\lambda)$, and the identity space for identity-based protocols as $\mathsf{IDSpace}(\lambda)$. All protocols have an implicit algorithm $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ that, given the security parameter, outputs public parameters pp. The public parameters are implicitly input to all subsequent algorithms. We write $x \leftarrow y$ to assign an expression y to variable x and $x \leftarrow\!\!\$\ S$ to sample uniformly from the set S. For a probabilistic algorithm A we denote by $y \leftarrow A(x; r)$ the execution of A with input x and randomness r, assigning the result to y. In short, we usually write $y \leftarrow\!\!\$\ A(x)$ to express the execution of A with uniform randomness. We denote the computational indistinguishability of two distributions X and Y as $X \approx_c Y$. For a positive integer n, we denote the set $\{1, \ldots, n\}$ as $[n]$. By $\{a_i\}_{i \in S}$ we denote the indexed set $\{(i, a_i) | i \in S\}$. We denote as $l_{i,S}(x)$ the Lagrange coefficient for interpolation among a set S at position x with $l_{i,S}(x) = \prod_{k \in S, k \neq i} \frac{x - x_k}{x_i - x_k}$. We write l_i instead when S is clear from context and $x = 0$. We assume that all shareholders in a (t, n)-threshold cryptosystem have participant indices in $[n]$.

Cryptography Building Blocks. We rely on three cryptographic building blocks, namely one-time signature schemes $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ and non-interactive zero-knowledge proofs of knowledge (NIZK-PoK) $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$. For the sake of brevity, we refer the reader to Appendix C.1 for detailed definitions. Additionally, we use hash functions in the random oracle model.

Bilinear Pairing Ensembles. Our constructions rely on *bilinear pairing ensembles*. A bilinear pairing ensemble $E = (\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ is an ensemble of cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ with generators g_1 and g_2 and $\mathbb{G}_T$ of prime order p along with a pairing operation

$e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies bilinearity and non-degeneracy. We rely on the Bilinear (Decisional) Diffie-Hellman (BDDH) [12] assumption as well as the XDH assumption [4, 10], which are both well-established and have been used in numerous works. We define them formally in Appendix C.2.

3 CCA-secure Traceable Threshold KEM

We first define our goal of CCA-secure Traceable Threshold KEM (TT−KEM). Then, in Section 3.2, we introduce some interesting technical modifications to the definitions of BT−KEM primitive to accommodate for CCA security. Finally, in Section 3.3, we show how to transform CCA-secure BT−KEM to CCA-secure TT−KEM.

3.1 Definition: TT−KEM with CCA-security

First, we introduce traceable threshold KEM, which was first defined in [15]. We make minor modifications to also define CCA-security.

Definition 3.1 (Traceable Threshold KEM). A traceable threshold KEM (TT−KEM) protocol is a tuple of algorithms Π = (KGen, Enc, Verify, PDec, Comb, Trace) with key space $\mathcal{K}(\lambda)$. It extends the standard threshold KEM by a tracing algorithm. The differences compared to standard threshold KEM are highlighted in gray.

- $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon(\lambda)})$. The KGen algorithm receives the number of parties n as well as the threshold t with $n \geq t > 0$ as well as an additional parameter ε, which is essentially a lower bound on the distinguishing advantage of decoders that must be traceable by this scheme (see Definitions C.8 and C.9). KGen outputs a public key pk, the secret key shares $\mathsf{sk} = \{\mathsf{sk}_i\}_{i\in[n]}$, and a tracing key tk.
- $(k, c, \tau) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk})$. Given a public key pk, the Enc algorithm returns a key $k \in \mathcal{K}(\lambda)$, a ciphertext c, and a tag τ.
- $1/0 \leftarrow \mathsf{Verify}(c, \tau)$. Given a ciphertext c and a tag τ, the Verify algorithm returns 1 if τ is valid for c and 0 otherwise.
- $d_i \leftarrow \mathsf{PDec}(\mathsf{sk}_i, c)$. The partial decryption algorithm receives a secret key share sk_i of party i and a ciphertext c as input and outputs a decryption share d_i.
- $k/\perp \leftarrow \mathsf{Comb}(S, \{d_i\}_{i\in S}, c)$. The Comb algorithm aggregates decryption shares d_i from a set of at least t parties S and outputs the key k encapsulated in the ciphertext c.
- $\mathcal{J} \leftarrow\!\$\ \mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^{1/\varepsilon(\lambda)})$. This algorithm receives a public key pk, a tracing key tk, the parameter ε, and black-box access to a decoder D. It outputs a set of traitors $\mathcal{J} \subseteq [n]$.

We put the correctness and IND−CCA definitions in Appendix C.3, but we note that they are the same as standard threshold KEM. In summary, correctness states that every honestly generated ciphertext decrypts correctly, using shares from any set of at least t parties. For IND−CCA security, we require that an adversary who controls up to $t-1$ parties is unable to distinguish if a challenge ciphertext belongs to a valid key or a random key from $\mathcal{K}(\lambda)$ instead. This needs to hold even if the adversary has access to a partial decryption oracle that allows him to learn partial decryptions on any ciphertext, but the challenge ciphertext.

Traceability. As in [15] traceability guarantees that if an adversary corrupts an arbitrary set of parties $\mathcal{J} \subseteq [n]$ (the traitors) and uses their shares to construct a decoder D that has decryption advantage ε, then the $\mathsf{Trace}^{D(\cdot)}$ algorithm, when given black-box access to the decoder, will output a nonempty subset $\mathcal{J}' \subseteq \mathcal{J}$ of traitors. It also captures unframeability that is, Trace's output $\mathcal{J}'$ cannot contain a non-traitor with non-negligible probability. Since we add tags τ for CCA-security, the decoder now also receives tags along with ciphertexts. The formal definition can be found in Appendix C.3 (Definition C.9).

3.2 Definition: BT−KEM with CCA-security

Next, we extend the BT−KEM definition from [15] to accommodate for CCA-security. Recall that in [15], the one-sided security notion was introduced in order to allow traceability. Hence, the main technical challenge is to achieve CCA-security while maintaining one-sided security of the BT−KEM scheme.

*Defining Bipartite Threshold Key Encapsulation (*BT−KEM*).* A natural way to define CCA security for BT−KEM is to let Enc algorithm output an additional tag τ to prove the integrity of the ciphertext. In our definition, we need to deviate from this approach to allow for schemes that offer both one-sided security and CCA security. More concretely, in the one-sided security game, the challenger generates two ciphertexts $c^{(i)} = (c_0^{(i)}, c_1^{(i)})$ for $i \in \{0,1\}$ and ensembles the challenge, either $c^* = (c_0^{(0)}, c_1^{(1)})$ when $b = 0$, or $c^* = (c_0^{(d)}, c_1^{(d)})$ when $b = 1$ (here $d \in \{0,1\}$ is chosen by the adversary). If we would let Enc output τ, then syntactically, we cannot receive tags for a malformed ciphertext c^* generated when $b = 0$.

To address this issue, we add algorithms to BT−KEM that allow us to generate tags both for valid ciphertexts during standard encryption and for malformed ciphertexts, as encountered in the one-sided security game. Concretely, we split encryption and tag generation into three distinct algorithms: TagKeys, Enc, and GenTag. The TagKeys algorithm generates a key pair $(\mathsf{sk}_e, \mathsf{vk}_e)$, where vk_e serves as a verification key. Encryption proceeds as $(k, (c_0, c_1)) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$[7]. Finally, GenTag uses the secret key sk_e to produce a tag $\tau \leftarrow\!\$\ \mathsf{GenTag}(c, \mathsf{sk}_e)$. Looking ahead, when constructing TT−KEM from BT−KEM, this mechanism is employed in two distinct ways:

(1) **Tags for Normal Encryption:** The three algorithms operate as described above. The encryptor first uses TagKeys to prepare a fresh key pair, then invokes Enc with vk_e to encrypt, and produces a tag within GenTag using sk_e.
(2) **Tags during Tracing with Malformed Ciphertexts:** Malformed ciphertext are constructed by mixing the left and right parts of two *different* ciphertexts (refer to the one-sided security game in Figure 3). To attach a valid tag, the tracer prepares a single key pair $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\!\$\ \mathsf{TagKeys}(1^\lambda)$. Then, it generates both $c^{(0)}$ and $c^{(1)}$ using two independent Enc operations, *but with the same* vk_e. Finally, the tracer uses GenTag to obtain a valid tag for the resulting

[7] Notice that the encryption takes vk_e as input. Conceptually, given $(\mathsf{sk}_e, \mathsf{vk}_e)$, one can generate a valid tag using sk_e for any ciphertext that was encrypted with the corresponding vk_e as additional input.

malformed ciphertext, given that both original ciphertexts were generated under the same vk_e that corresponds to sk_e.

Additionally, we add a Verify algorithm to validate ciphertext-tag pairs prior to partial decryption. In practice, shareholders will only release partial decryptions for ciphertexts with valid tags.

The requirement that tags, as generated in normal encryption, verify is captured by two-sided correctness, while the latter requirement that tags verify for malformed ciphertexts is implicitly captured by our modified one-sided security (Figure 3) and two-sided correctness. In particular, one-sided security also requires that the additional tag does not gain the adversary any advantage in distinguishing whether it receives a malformed ciphertext or not.

*Definition 3.2 (Bipartite Threshold Key Encapsulation(*BT−KEM*)).* A BT−KEM scheme is a tuple of algorithms Π = (KGen, TagKeys, Enc, GenTag, Verify, PDec, Comb) with key space $\mathcal{K}(\lambda)$. The necessary modifications of BT−KEM to achieve CCA-Security are highlighted in gray.

- $(\mathsf{pk}, \{(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})\}_{i\in[n],j\in[\ell]}) \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, \ell)$. KGen receives the security parameter λ, the number of parties n, a threshold t such that $0 < t \leq n$, and the number of positions $\ell \in \mathbb{N}$. It outputs a public key pk and, for every $(i, j) \in [n] \times [\ell]$, secret shares $(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})$. We call $\mathsf{sk}_{i,0}^{(j)}$ a *left* key share and $\mathsf{sk}_{i,0}^{(j)}$ a *right* key share. As a shorthand, we denote the set of all secret key shares as sk and all secret key shares for party i as sk_i. Further, $\mathsf{sk}_{i,0}$ denotes all the left shares in sk_i while $\mathsf{sk}_{i,1}$ denotes all the right shares in sk_i.
- $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\$ \mathsf{TagKeys}(1^\lambda)$. This algorithm outputs a key pair $(\mathsf{sk}_e, \mathsf{vk}_e)$ that is used to generate tags for ciphertexts.
- $(k, c \coloneqq (c_0, c_1)) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$. The Enc algorithm gets the public key pk, a position $j \in [\ell]$, and a verification key vk_e. It outputs a key $k \in \mathcal{K}(\lambda)$ and a ciphertext $c \coloneqq (c_0, c_1)$.
- $\tau \leftarrow\$ \mathsf{GenTag}(c, \mathsf{sk}_e)$. The GenTag algorithm receives a ciphertext c and a key sk_e. It outputs a tag τ.
- $1/0 \leftarrow \mathsf{Verify}(c, \tau)$. Verify outputs 1, iff the tag is valid for c.
- $d_i \leftarrow \mathsf{PDec}(\mathsf{sk}_{i,b_{i,j}}^{(j)}, c)$. The partial decryption algorithm receives a left ($b_{i,j} = 0$) or right ($b_{i,j} = 1$) secret share of i for a position $j \in [\ell]$ and a ciphertext c. It outputs a partial decryption d_i.
- $k/\perp \leftarrow \mathsf{Comb}(S, \{d_i\}_{i\in S}, c)$. Given decryption shares d_i from a set of at least t parties S and a ciphertext c, the Comb algorithm returns the KEM key k or $\perp$ upon failure.

Definition 3.3 (Two-Sided Correctness of BT−KEM*).* A BT−KEM protocol is two-sided correct if for all λ, $\ell > 0$, all $j \in [\ell]$ and $n \geq t > 0$ as well as all $S \in [n]$ with $|S| > t$ and all bitstrings $b \in \{0,1\}^n \times \{0,1\}^\ell$ it holds that

$$\Pr\left[\begin{array}{c} k = k' \wedge \\ \mathsf{Verify}(c,\tau) = 1 \end{array} : \begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, \ell) \\ (\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\$ \mathsf{TagKeys}(1^\lambda) \\ (k, c) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, j, \mathsf{vk}_e) \\ \tau \leftarrow\$ \mathsf{GenTag}(c, \mathsf{sk}_e) \\ \forall i \in S\colon d_i \leftarrow \mathsf{PDec}(\mathsf{sk}_{i,b_{i,j}}^{(j)}, c) \\ k' \leftarrow \mathsf{Dec}(S, \{d_i\}_{i\in S}, c) \end{array}\right] = 1.$$

3.2.1 CCA and One-Sided Security. Next, we define the required security games for modeling CCA-security in BT−KEM. First, we define a CCA-security game. Then, we present a one-sided security game. Our one-sided security game follows [15] but with slight modifications that are needed to achieve CCA-security.

CCA-security Game. In Game−IND−CCA (Figure 2), the adversary can corrupt a set of up to $t - 1$ parties C. We also give the adversary access to a partial decryption oracle O^{PDec} that derives the left and right partial decryptions for ciphertexts on behalf of honest parties[8]. The only restriction is that the adversary can not use O^{PDec} to obtain more than $t - 1$ decryption shares on the challenge ciphertext c^* (counting the shares the adversary can trivially compute through corrupted parties in C). Additionally, the challenge ciphertext now also contains the tag τ^*. The adversary wins if it can successfully distinguish between a random KEM-key and the one encapsulated in c^*.

$\mathsf{Game-IND-CCA}_{\mathcal{A}}(1^\lambda)$	$O^{\mathsf{PDec}}((c,\tau), i)$
$\mathsf{ctr} \leftarrow 0; b \leftarrow\$ \{0,1\}$	**if** $\mathsf{Verify}(c,\tau) = 0$ **then**
$(\ell, \mathsf{st}_1) \leftarrow\$ \mathcal{A}_1(1^\lambda)$	**return** $\perp$
$(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, \ell)$	**if** $c = c^*$ **then**
$(C, \mathsf{st}_2) \leftarrow\$ \mathcal{A}_2(\mathsf{st}_1, \mathsf{pk})$	$\mathsf{ctr} \leftarrow\$ \mathsf{ctr} + 1$
if $C \not\subseteq [n] \vee \lvert C\rvert \geq t$ **then return** 0	**if** $\mathsf{ctr} \geq t - \lvert C\rvert$ **then**
$j^* \leftarrow\$ [\ell]$; $k_0 \leftarrow\$ \mathcal{K}(\lambda)$	**return** $\perp$
$(\mathsf{sk}_e^*, \mathsf{vk}_e^*) \leftarrow\$ \mathsf{TagKeys}(1^\lambda)$	$d_{i,0} \leftarrow \mathsf{PDec}(c, \mathsf{sk}_{i,0})$
$(k_1, c^*) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, j^*, \mathsf{vk}_e^*)$	$d_{i,1} \leftarrow \mathsf{PDec}(c, \mathsf{sk}_{i,1})$
$\tau^* \leftarrow\$ \mathsf{GenTag}(c^*, \mathsf{sk}_e^*)$	**return** $(d_{i,0}, d_{i,1})$
$b' \leftarrow\$ \mathcal{A}_3^{O^{\mathsf{PDec}}}(\mathsf{st}_2, (c^*, \tau^*), k_b), \{\mathsf{sk}_i\}_{i\in C})$	
return $b \stackrel{?}{=} b'$	

Figure 2: IND−CCA **security game for** BT−KEM**. The changes to the IND-CPA game in [15] are highlighted in gray.**

*Definition 3.4 (*IND−CCA *Security of* BT−KEM*).* A BT−KEM protocol achieves IND−CCA security, if for all PPT adversaries $\mathcal{A} \coloneqq (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\mathsf{Game-IND-CCA}_{\mathcal{A}}^{\mathsf{BT-KEM}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where Game−IND−CCA is defined in Figure 2.

Modified One-Sided Security Game. We define one-sided security similarly to [15], but account for the CCA-tags that are attached to ciphertexts. The adjusted Game−OSS experiment (Figure 3) first prepares a key pair $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\$ \mathsf{TagKeys}(1^\lambda)$, assembles the challenge ciphertext c^* as in [15], and finally uses sk_e to generate a tag τ^* for c^* as $\tau^* \leftarrow\$ \mathsf{GenTag}(c^*, \mathsf{sk}_e)$. We give τ^* to the adversary; thus in our security analysis, we need to show that the tag does not help the adversary to distinguish whether it receives a valid ciphertext (case $b = 0$) or malformed ciphertext (case $b = 1$). We

[8]Note that in normal decryptions, the parties contribute either a left or a right share, not both.

recall that the "natural" approach of generating τ^* within Enc does not work even on a definition level: If the two encryptions to ciphertexts $c^{(0)}$ and $c^{(1)}$ generated in Game−OSS would generate separate tags $\tau^{(0)}$ and $\tau^{(1)}$, then it is unclear which tag should be given to the adversary $\mathcal{A}_2$ together with the challenge ciphertext c^*. Moreover, these tags are not valid for a mixed ciphertext c^* (i.e., when $b = 0$).

$\mathsf{Game{-}OSS}_{\mathcal{A}}(1^\lambda)$

$(\ell, u, d, \mathsf{st}_1) \leftarrow\!\$\ \mathcal{A}_1(1^\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda, n, t, \ell)$

$(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\!\$\ \mathsf{TagKeys}(1^\lambda)$

$(k^{(0)}, c^{(0)} \coloneqq (c_0^{(0)}, c_1^{(0)})) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, u, \mathsf{vk}_e)$

$(k^{(1)}, c^{(1)} \coloneqq (c_0^{(1)}, c_1^{(1)})) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, u, \mathsf{vk}_e)$

$b \leftarrow\!\$\ \{0, 1\}$

if $b = 0$ **then** $c^* \leftarrow (c_0^{(0)}, c_1^{(1)})$

if $b = 1$ **then** $c^* \leftarrow (c_0^{(d)}, c_1^{(d)})$

$\tau^* \leftarrow\!\$\ \mathsf{GenTag}(c^*, \mathsf{sk}_e)$

$\mathsf{shares} \leftarrow \left(\{(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})\}_{i\in[n], j\in[\ell]\setminus\{u\}}, \{\mathsf{sk}_{i,d}^{(u)}\}_{i\in[n]}\right)$

$b' \leftarrow\!\$\ \mathcal{A}_2(\mathsf{st}_1, \mathsf{pk}, (c^*, \tau^*), k^{(0)}, \mathsf{shares})$

return $b \stackrel{?}{=} b'$

Figure 3: One-sided security game for BT−KEM. The modifications compared to [15] are highlighted in gray.

Definition 3.5 (One-sided Security of BT−KEM). A BT−KEM protocol achieves one-sided security, if for all PPT adversaries $\mathcal{A} \coloneqq (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\mathsf{Game{-}OSS}_{\mathcal{A}}^{\mathsf{BT-KEM}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where Game−OSS is defined in Figure 3.

3.3 Generic Construction: CCA BT−KEM to CCA TT−KEM

In this section, we show that plugging any CCA-secure BT−KEM, as presented in Section 3.2, in the black-box TT−KEM construction of [15] results in a CCA-secure TT−KEM (later, in Section 4.2 we construct a CCA-secure BT−KEM). We need to apply some minor modifications to TT−KEM in order to account for the changes we made to BT−KEM. Next, we will overview the transformation and discuss our modifications.

The BT−KEM to TT−KEM Transformation [15]. The transformation samples a fingerprinting code $\Gamma_{n,\ell}$ of n codewords of size ℓ, sets the tracing key tk to be the tracing key in Γ, and then receives the secret shares from BT−KEM, where the number of positions is set to ℓ (i.e., the size of each codeword). Each party is assigned a codeword, and the secret shares are distributed according to the corresponding codeword; if the j'th bit of the i'th codeword is 0, then party i receives the left share at position j, and the right share otherwise. Hence, each party receives exactly one share for every position $j \in [\ell]$. In encryption, TT−KEM.Enc samples a position $j \leftarrow\!\$\ [\ell]$ at random and invokes BT−KEM.Enc(pk, j) with the sampled position.

In our case, we adjust how TT−KEM.Enc uses BT−KEM for encryption. That is, instead of simply invoking BT−KEM.Enc(pk, j) with a random $j \in [\ell]$, it should first prepare a key pair $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\!\$\ \mathsf{TagKeys}(1^\lambda)$, then encrypt $(k, c) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$ for a random $j \in [\ell]$, and finally compute the tag $\tau \leftarrow\!\$\ \mathsf{GenTag}(c, \mathsf{sk}_e)$. The resulting encryption algorithm outputs the pair (c, π). Also, we set TT−KEM.Verify to BT−KEM.Verify so shareholders can check the tag's validity before releasing partial decryptions on a ciphertext.

Note the resulting TT−KEM scheme inherits its IND−CCA security directly from that of our CCA-secure BT−KEM. We next discuss traceability.

Modifications to Tracing. We modify the Tr subroutine of the tracing algorithm in order to generate tags for the malformed ciphertexts. The new Tr subroutine is shown in Figure 4. Observe that the modified Tr subroutine generates the tag in exactly the same manner as for the challenge ciphertext in the modified one-sided security game (Figure 3). The Trace algorithm is otherwise unchanged from that of [15]. Intuitively, it uses the Tr subroutine to determine for which positions $j \in [\ell]$ the decoder has only left or only right keys. To do so, it queries the Tr subroutine for every position j with three different configurations: A completely valid ciphertext ($(b_k, b_0, b_1) = (1, 1, 1)$), a valid left ciphertext and invalid right ciphertext ($(b_k, b_0, b_1) = (0, 0, 1)$), and an invalid left and right ciphertext ($(b_k, b_0, b_1) = (1, 0, 0)$). In the end, the collected information over all ℓ positions is used to determine the traitors by invoking the trace function of the underlying fingerprinting code. More details can be found in Appendix E.3.

$\mathsf{Tr}^{D(\cdot)}(\mathsf{pk}, j, N, (b_k, b_0, b_1))$

$\mathsf{ctr} \leftarrow 0$

for $r = 1, \ldots, N$ **do**

  $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\!\$\ \mathsf{TagKeys}(1^\lambda)$

  $(k^{(0)}, c^{(0)} \coloneqq (c_0^{(0)}, c_1^{(0)})) \leftarrow\!\$\ \mathsf{BT{-}KEM.Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$

  $(k^{(1)}, c^{(1)} \coloneqq (c_0^{(1)}, c_1^{(1)})) \leftarrow\!\$\ \mathsf{BT{-}KEM.Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$

  $c^* \leftarrow (c_0^{(b_0)}, c_1^{(b_1)})$

  $\tau^* \leftarrow\!\$\ \mathsf{GenTag}(c^*, \mathsf{sk}_e)$

  if $D((c^*, \tau^*), k^{(b_k)}) = 1$ **then** $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

return ctr

Figure 4: The adjusted Tr subroutine of Trace. Changes compared to that of [15] are highlighted in gray.

Intuitively, our modifications to Tr do not break traceability because our modified one-sided security notion guarantees that for a secure BT−KEM, the additional tag τ^* does not help the adversary to distinguish malformed ciphertexts from normal ciphertexts. We refer to Appendix E.3 for a detailed overview of how the proof of traceability in [15] needs to be adjusted for our needs.

Corollary 3.6. *Let* Π *be a* BT−KEM *protocol with two-sided correctness (Definition 3.3), CCA-security (Definition 3.4), and one-sided*

security (Definition 3.5). Applying the traceability transformation of [15] to Π *with the above changes to* Enc, Verify, *and* Tr *yields a CCA-secure* TT−KEM *as defined in Section 3.1.*

4 CCA BT−KEM from Identity-Based BT−KEM

In this section, we introduce our main building block of Bipartite Threshold Identity-based KEM (BTIB−KEM). It is central to our results, as we use it both to construct CCA-secure BT−KEM (and CCA-secure TT−KEM as a consequence) and traceable threshold IBE. We start with a new definition of BTIB−KEM and show how to generically transform that to CCA-secure BT−KEM in Section 4.2.

4.1 Definition: Bipartite Threshold Identity Based Key Encapsulation (BTIB−KEM)

Our BTIB−KEM definition is an adaptation of the bipartite threshold KEM definition of [15] to the identity-based setting.

Definition 4.1 (Bipartite Threshold Identity Based Key Encapsulation). A BTIB−KEM protocol is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{DIdk}, \mathsf{CombIdk}, \mathsf{Dec})$ with key space $\mathcal{K}(\lambda)$ and identity space $\mathsf{IDSpace}(\lambda)$ such that:

- $(\mathsf{pk}, \{(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})\}_{i\in[n],j\in[\ell]}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda, n, t, \ell)$. This algorithm has the same syntax as BT−KEM.KGen (Definition 3.2).
- $(k, c \coloneqq (c_0, c_1)) \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, j, \mathsf{ID})$. Given a public key pk, index $j \in [\ell]$, and an identity $\mathsf{ID} \in \mathsf{IDSpace}(\lambda)$, Enc outputs an encapsulation key $k \in \mathcal{K}(\lambda)$ and an encryption of k to identity ID. The ciphertext $c = (c_0, c_1)$, is split into a *left ciphertext* c_0 and *right ciphertext* c_1.
- $\mathsf{idk}_i \leftarrow \mathsf{DIdk}(\{\mathsf{sk}_{i,b_{i,j}}^{(j)}\}_{j\in[\ell]}, \mathsf{ID})$. The identity key derivation algorithm DIdk receives i's secret key share $\mathsf{sk}_{i,b_{i,j}}^{(j)}$ for every index $j \in [\ell]$. If $b_{i,j} = 0$, then the given share in position j is the left share. Otherwise, it is the right share. DIdk outputs an identity key share idk_i for identity ID. We also define a version of this algorithm that only derives the identity-key share for position j as $\mathsf{idk}_i^{(j)} \leftarrow \mathsf{DIdk}_j(\mathsf{sk}_{i,b_{i,j}}^{(j)}, \mathsf{ID}, j)$.
- $\mathsf{idk} \leftarrow \mathsf{CombIdk}(S, \{\mathsf{idk}_i\}_{i\in S})$. The CombIdk algorithm combines at least t identity key shares idk_i from parties $i \in S$ where $S \in [n]$ and $|S| \geq t$ into a full identity key idk. Again, we define a version of CombIdk specifically for position j as $\mathsf{idk}^{(j)} \leftarrow \mathsf{CombIdk}(S, \{\mathsf{idk}_i^{(j)}\}_{i\in S})$.
- $k/\perp \leftarrow \mathsf{Dec}(\mathsf{idk}, c, j)$. Given an identity key idk, a ciphertext $c = (c_0, c_1)$, and an index $j \in [\ell]$, Dec decrypts c and returns the encapsulation key k or $\perp$ in case of failure.

A BTIB−KEM must satisfy two-sided correctness, semantic security, and one-sided security.

Definition 4.2 (Two-Sided Correctness of BTIB−KEM*).* We say that a BTIB−KEM protocol is two-sided correct if for all $\lambda, \ell > 0$, $j \in [\ell]$ and $n \geq t > 0$ as well as all $S \in [n]$, where $|S| \geq t$, all bitstrings $b \in \{0,1\}^n \times \{0,1\}^\ell$ and all identities $\mathsf{ID} \in \mathsf{IDSpace}(\lambda)$ it holds that

$$\Pr\left[k = k' : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda, n, t, \ell)\\ (k, c) \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, j, \mathsf{ID})\\ \forall i \in S\colon \mathsf{idk}_i \leftarrow \mathsf{DIdk}(\{\mathsf{sk}_{i,b_{i,j}}^{(j)}\}_{j\in[\ell]}, \mathsf{ID})\\ \mathsf{idk} \leftarrow \mathsf{CombIdk}(S, \{\mathsf{idk}_i\}_{i\in S})\\ k' \leftarrow \mathsf{Dec}(\mathsf{idk}, c, j)\end{array}\right] = 1.$$

Semantic Security. For semantic security, we work in the selective identity model [11, 22], where the adversary is required to commit to the identity it intends to attack ahead of time. While this is a relaxation, we later show that *selective-identity* CPA-secure BTIB−KEM is sufficient for our transformation to BT−KEM to achieve CCA-security. Both of our BTIB−KEM constructions can be proven without selective identity using standard techniques (i.e. guessing the challenge identity in the random oracle with a tightness loss in the number of random oracle queries q_H [7]).

$\mathsf{Game-IND-SID-CPA}_{\mathcal{A}}(1^\lambda)$	$\mathcal{O}^{\mathsf{DIdk}}(\mathsf{ID}, i)$
$\mathsf{ctr} \leftarrow 0; b \leftarrow\!\!\$\ \{0,1\}$	**if** $\mathsf{ID} = \mathsf{ID}^*$ **then**
$(\mathsf{ID}^*, \ell, \mathsf{st}_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$	$\quad \mathsf{ctr} \leftarrow\!\!\$\ \mathsf{ctr} + 1$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda, n, t, \ell)$	$\quad$ **if** $\mathsf{ctr} \geq t - \lvert C\rvert$ **then**
$(C, \mathsf{st}_2) \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{st}_1, \mathsf{pk})$	$\quad\quad$ **return** $\perp$
if $C \not\subseteq [n] \vee \lvert C\rvert \geq t$ **then return** 0	$\mathsf{idk}_0 \leftarrow \mathsf{DIdk}(\mathsf{sk}_{i,0}, \mathsf{ID})$
$j^* \leftarrow\!\!\$\ [\ell];\ k_0 \leftarrow\!\!\$\ \mathcal{K}(\lambda)$	$\mathsf{idk}_1 \leftarrow \mathsf{DIdk}(\mathsf{sk}_{i,1}, \mathsf{ID})$
$(k_1, c) \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, j^*, \mathsf{ID}^*)$	**return** $(\mathsf{idk}_0, \mathsf{idk}_1)$
$b' \leftarrow\!\!\$\ \mathcal{A}_3^{\mathcal{O}^{\mathsf{DIdk}}}(\mathsf{st}_2, (c, k_b), \{\mathsf{sk}_i\}_{i\in C})$	
return $b \stackrel{?}{=} b'$	

Figure 5: IND−SID−CPA **security game for** BTIB−KEM**.**

Definition 4.3 (Selective ID Semantic Security). A BTIB−KEM protocol achieves selective-identity semantic security (IND−SID−CPA), if for all PPT adversaries $\mathcal{A} \coloneqq (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\mathsf{Game-IND-SID-CPA}_{\mathcal{A}}^{\mathsf{BTIB-KEM}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where Game−IND−SID−CPA is defined in Figure 5.

One-Sided Security. Following [15], we define one-sided security for BTIB−KEM through Game−OSS−ID (Figure 6), but in the identity-based setting. When we later use bipartite identity-based key encapsulation as a building block for traceable threshold identity-based KEM, we will rely on one-sided security of BTIB−KEM to ensure that we can identify a set of traitors. Roughly speaking, we will use one-sided security to argue that if a decoder has only left keys ($d = 0$) for some index u, then it can not distinguish, given $k, (c_0, c_1)$, whether c_1 fits c_0 and k or not. Importantly, if BTIB−KEM lacks this property, then a decoder could avoid being traced at all.

Observe that the one-sided security game does not provide an identity key derivation oracle. This is reasonable, as the adversary

$\mathsf{Game{-}OSS{-}ID}_{\mathcal{A}}(1^\lambda)$

$(\ell, u, d, \mathsf{ID}, \mathsf{st}_1) \leftarrow\$ \mathcal{A}_1(1^\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, \ell)$

$(k^{(0)}, c^{(0)} \coloneqq (c_0^{(0)}, c_1^{(0)})) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, u, \mathsf{ID})$

$(k^{(1)}, c^{(1)} \coloneqq (c_0^{(1)}, c_1^{(1)})) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, u, \mathsf{ID})$

$b \leftarrow\$ \{0, 1\}$

if $b = 0$ **then** $c^* \leftarrow (c_0^{(0)}, c_1^{(1)})$

if $b = 1$ **then** $c^* \leftarrow (c_0^{(d)}, c_1^{(d)})$

$\mathsf{shares} \leftarrow \left(\{(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})\}_{i\in[n], j\in[\ell]\setminus\{u\}}, \{\mathsf{sk}_{i,d}^{(u)}\}_{i\in[n]}\right)$

$b' \leftarrow\$ \mathcal{A}_2(\mathsf{st}_1, \mathsf{pk}, c^*, k^{(0)}, \mathsf{shares})$

return $b \stackrel{?}{=} b'$

Figure 6: One-sided security game for BTIB−KEM.

obtains sufficient information to derive any identity key and, in consequence, decrypt any ciphertext using the master secret key shares he is provided in Game−OSS−ID. One could imagine a stronger definition of one-sided security, where, e.g., the adversary who only obtains left keys for u gets to see *partial right-side identity-keys*, provided they are for a different identity than the challenge identity. This oracle, however, is not required to prove our transformation to CCA-secure BT−KEM. We leave it as an open problem to find constructions that can provably satisfy identity-based one-sided security with an additional identity key derivation oracle. As a direct implication, one would get even stronger traceability guarantees for TTIB−KEM and TT−KEM. We elaborate on this in Appendix D.1.

Definition 4.4 (One-sided security of BTIB−KEM*).* A BTIB−KEM protocol achieves one-sided security, if for all PPT adversaries $\mathcal{A} \coloneqq (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\mathsf{Game{-}OSS{-}ID}_{\mathcal{A}}^{\mathsf{BTIB-KEM}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where Game−OSS−ID is defined in Figure 6.

Looking ahead, we will show how to construct our main building block of BTIB−KEM in Section 5.

4.2 Generic Construction: BTIB−KEM to CCA-Secure BT−KEM

In this section, we present a black-box CCA-secure BT−KEM construction. The underlying schemes in this construction are a CPA-secure BTIB−KEM and a one-time signature scheme.

Let Σ = (KGen, Sign, Verify) be a one-time signature scheme that is EUF−OT−CMA-secure (Definition C.2) and let $\mathcal{I}$ be a BTIB−KEM that fulfills semantic security (Definition 4.3) and one-sided security (Definition 4.4).

We construct CCA-secure BT−KEM by adapting the transformation from CPA-IBE to CCA-secure encryption [11] to the bipartite setting while ensuring that our adjusted notion of one-sided security holds. The construction is depicted in Figure 7.

In our transformation, the encryptor samples a fresh signing keypair $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\$ \Sigma.\mathsf{KGen}(1^\lambda)$ per encryption operation. Then it calls $(k, c_0, c_1) \leftarrow\$ \mathcal{I}.\mathsf{Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$, where it encrypts to the identity "vk_e". The encryptor assembles $c \coloneqq ((k, c_0, c_1), \mathsf{vk}_e, j)$ and finally calls GenTag to sign the entire ciphertext under sk_e. The signature σ is attached as CCA-tag to c. The Verify algorithm verifies σ for c under vk_e, and should be called prior to partial decryption to ensure CCA-security. During partial decryption of a ciphertext $c = ((k, c_0, c_1), \mathsf{vk}_e, j)$, the partial identity-key for "vk_e" is derived using $\mathcal{I}.\mathsf{DIdk}$. One can run Comb to combine at least t partial identity keys for vk_e and subsequently decrypt c. We highlight that it is sufficient to derive the j'th position of the identity key during PDec. Hence, the partial decryptions do not depend on the number of positions ℓ.

$\mathsf{KGen}(1^\lambda, n, t, \ell)$

return $\mathcal{I}.\mathsf{KGen}(1^\lambda, n, t, \ell)$

$\mathsf{TagKeys}(1^\lambda)$

$(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\$ \Sigma.\mathsf{KGen}(1^\lambda)$

return $(\mathsf{sk}_e, \mathsf{vk}_e)$

$\mathsf{Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$

$\mathsf{ID} \leftarrow \mathsf{vk}_e$

$(k, c_0, c_1) \leftarrow\$ \mathcal{I}.\mathsf{Enc}(\mathsf{pk}, j, \mathsf{ID})$

return $k, c \coloneqq ((c_0, c_1), j, \mathsf{vk}_e)$

$\mathsf{GenTag}(c, \mathsf{sk}_e)$

$\sigma \leftarrow\$ \Sigma.\mathsf{Sign}(\mathsf{sk}_e, c)$

return $\tau \coloneqq \sigma$

$\mathsf{Verify}(c, \tau)$

Parse c as $((c_0, c_1), j, \mathsf{vk}_e)$

and τ as σ

return $\Sigma.\mathsf{Verify}(\mathsf{vk}_e, c, \sigma)$

$\mathsf{PDec}(\mathsf{sk}_{i,b_{i,j}}^{(j)}, c)$

$\mathsf{ID} \leftarrow \mathsf{vk}_e$

$\mathsf{idk}_i^{(j)} \leftarrow \mathcal{I}.\mathsf{DIdk}_j(\mathsf{sk}_{i,b_{i,j}}^{(j)}, \mathsf{ID}, j)$

return $d_i \coloneqq \mathsf{idk}_i^{(j)}$

$\mathsf{Comb}(S, \{d_i\}_{i\in S}, c)$

Parse $d_i \coloneqq \mathsf{idk}_i^{(j)}; c \coloneqq ((c_0, c_1), j, \mathsf{vk}_e)$

$\mathsf{idk} \leftarrow \mathcal{I}.\mathsf{CombIdk}(S, \{\mathsf{idk}_i^{(j)}\}_{i\in S})$

return $\mathcal{I}.\mathsf{Dec}(\mathsf{idk}, (c_0, c_1), j)$

Figure 7: Black-box CCA-secure BT−KEM **from any** BTIB−KEM **scheme $\mathcal{I}$ and signature scheme Σ.**

Correctness and Security. The resulting BT−KEM clearly has two-sided correctness, given the two-sided correctness of BTIB−KEM and the correctness of the one-time signature scheme Σ.

For IND−CCA-security, notice that the challenge ciphertext is encrypted to the identity "vk_e" in $\mathcal{I}$. Due to the semantic security of $\mathcal{I}$, the adversary can not learn any information about the challenge ciphertext on its own. Furthermore, the decryption oracle O^{PDec} does not give the adversary any advantage. Indeed, if the queried ciphertext contains another vk_e, then O^{PDec} derives shares of a different identity key. If it contains the same vk_e then the adversary would need to forge a signature under vk_e, which is infeasible according to the unforgeability of Σ. A detailed proof for the following lemma is presented in Appendix E.1.

Lemma 4.5 (IND−CCA Security of BT−KEM). *The* BT−KEM *construction as presented in Figure 7 is* IND−CCA *secure if the underlying* BTIB−KEM *is* IND−SID−CPA *secure and the signature scheme is* EUF−OT−CMA *secure.*

Lemma 4.6 (One-sided Security of BT−KEM). *The* BT−KEM *construction as presented in Figure 7 is one-sided secure (Definition 3.5) if the underlying* BTIB−KEM *is one-sided secure (Definition 4.4).*

The one-sided security reduces straightforwardly to one-sided security of BTIB−KEM. The detailed proof is given in Appendix E.2

5 Concrete Instantiations of BTIB−KEM

Both our results for CCA-secure TT−KEM (Section 3.3) and traceable threshold IBE (Appendix D) rely on the existence of efficient constructions for our main building block of bipartite threshold identity-based KEM (BTIB−KEM). To this end, we present two constructions based on bilinear pairings within this section. Henceforth, we refer to them as BTIB−KEM−1 and BTIB−KEM−2.

In both of our constructions, we combine techniques used in [15] with an appropriate IBE scheme. In particular, the constructions of [15] are based on the following idea. First, generate two secret keys, y, z, for the left and right sides. In order to achieve two-sided correctness, a correlation must exist between the two sides. This correlation is achieved via a master secret α. That is, the left secret shares correspond to αz, and the right secret shares correspond to αy. The encapsulated key is then $e(g_1, g_2)^{\alpha yzr}$, where r is sampled in the encryption. The one-sided security is guaranteed by the argument that αz looks random to parties that hold shares of αy, and vice versa. In BTIB−KEM, we need to accommodate for ℓ secret keys for every side. In our first construction, we treat each one of the ℓ instances independently, which results in a public key of size ℓ. We improve this in our second construction by using a 2-Layer IBE, but at the cost of a longer ciphertext (i.e., 12 vs 2 group elements) and more complicated construction. Interestingly, as we already discussed, our second construction can be easily extended to multi-layer IBE using techniques from [7], which allows us to achieve CCA-secure traceable IBE.

In Appendix D.1, we define TTIB−KEM and show a TTIB−KEM black-box construction from BTIB−KEM, which can be instantiated with both BTIB−KEM−1 and BTIB−KEM−2.

5.1 BTIB−KEM−1 (Short Ciphertext)

Our first BTIB−KEM construction (BTIB−KEM−1), depicted in Figure 8, is based on Boneh-Franklin IBE [12].

Basically, we extend Boneh-Franklin IBE to achieve the requirements of two-sided correctness and one-sided security.

Setup and Key Generation. The BTIB−KEM−1 Setup generates an asymmetric pairing ensemble E as public parameters. The key space of this construction is the target group $\mathcal{K}(\lambda) = \mathbb{G}_T$ while the identity space is arbitrary. Further, we also use a cryptographic hash function $\mathcal{H}$ with output domain $\mathbb{G}_2$. During KGen we sample α_j, y_j and z_j for every $j \in [\ell]$. Intuitively, this is the equivalent of ℓ separate Boneh-Franklin IBE instances with $\alpha_j y_j z_j$ as master secret key and $X_j = g_1^{\alpha_j y_j z_j}$ as public key. For every j, we share this master key into left and right key shares by generating t-out-of-n shares of α_j, denoted by $\{s_i^{(j)}\}_{i\in[n]}$. Then, the left share party i gets for index j is $z_j \cdot s_i^{(j)}$ and the right share is $y_j \cdot s_i^{(j)}$.

Encryption. For encryption and identity key derivation, we follow Boneh-Franklin IBE and extend it to work with correlated keys. For encryption to a specific $j \in [\ell]$ and an identity ID, one generates the KEM key as $k \leftarrow e(X_j^r, \mathcal{H}(j, \mathsf{ID}))[= e(g_1, \mathcal{H}(j, \mathsf{ID}))^{\alpha_j y_j z_j r}]$ for a random $r \leftarrow\!\!\$\ \mathbb{Z}_p$. Instead of a single ciphertext as g_1^r, we separate this into a left and right ciphertext $(c_0, c_1) \leftarrow (Y_j^r, Z_j^r)$. While during key generation, the left shares contain z_j, here, the left ciphertext contains y_j. This allows us to combine those parts of the identity key that stem from left shares with the left part of the ciphertext and vice versa in order to decrypt.

$\mathsf{KGen}(1^\lambda, n, t, \ell)$

for $j \in [\ell]$ **do**
- $\alpha_j, y_j, z_j \leftarrow\!\!\$\ \mathbb{Z}_p$
- $(X_j, Y_j, Z_j) \leftarrow (g_1^{\alpha_j y_j z_j}, g_1^{y_j}, g_1^{z_j})$
- $\{s_i^{(j)}\}_{i\in[n]} \leftarrow\!\!\$\ \mathsf{Shamir.Share}(\alpha_j, n, t)$
- **for** $i \in [n]$ **do**
  - $\mathsf{sk}_{i,0}^{(j)} \leftarrow z_j \cdot s_i^{(j)}$
  - $\mathsf{sk}_{i,1}^{(j)} \leftarrow y_j \cdot s_i^{(j)}$

$\mathsf{pk} \leftarrow \{(X_j, Y_j, Z_j)\}_{j\in[\ell]}$

$\mathsf{sk} \leftarrow \{(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})\}_{j\in[\ell], i\in[n]}$

return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{DIdk}(\{\mathsf{sk}_{i,b_{i,j}}^{(j)}\}_{j\in[\ell]}, \mathsf{ID})$

for $j \in [\ell]$ **do**
- $\mathsf{idk}_i^{(j)} \leftarrow \mathcal{H}(j, \mathsf{ID})^{\mathsf{sk}_{i,b_{i,j}}^{(j)}}$

return $\mathsf{idk}_i \coloneqq \{(b_{i,j}, \mathsf{idk}_i^{(j)})\}_{j\in[\ell]}$

$\mathsf{Enc}(\mathsf{pk}, j, \mathsf{ID})$

$r \leftarrow\!\!\$\ \mathbb{Z}_p$

$k \leftarrow e(X_j^r, \mathcal{H}(j, \mathsf{ID}))$

$(c_0, c_1) \leftarrow (Y_j^r, Z_j^r)$

return $(k, (c_0, c_1))$

$\mathsf{CombIdk}(S, \{\mathsf{idk}_i\}_{i\in S})$

for $j \in [\ell]$ **do**
- $S_0 \leftarrow \{i \in S \mid b_{i,j} = 0\}$
- $S_1 \leftarrow S \setminus S_0$
- $L_j \leftarrow \prod_{i\in S_0} \mathsf{idk}_i^{(j)\, l_{i,S}}$
- $R_j \leftarrow \prod_{i\in S_1} \mathsf{idk}_i^{(j)\, l_{i,S}}$

return $\mathsf{idk} \coloneqq \{(L_j, R_j)\}_{j\in[\ell]}$

$\mathsf{Dec}(\mathsf{idk}, c = (c_0, c_1), j)$

$k \leftarrow e(c_0, L_j) \cdot e(c_1, R_j)$

return k

Figure 8: Our BTIB−KEM−1 Construction.

Identity Key Derivation. Party i calls DIdk to derive the key share for ID and provides a left or right key share (the side of the share is determined by the bit $b_{i,j}$). The identity key share would be $\mathcal{H}(j, \mathsf{ID})^{\mathsf{sk}_{i,b_{i,j}}^{(j)}}$. That is, party i sends to the entity with ID the share $\mathcal{H}(j, \mathsf{ID})^{z_j \cdot s_i^{(j)}}$ or $\mathcal{H}(j, \mathsf{ID})^{y_j \cdot s_i^{(j)}}$. Using CombIdk, for every $j \in [\ell]$, one can locally aggregate t or more identity key shares from a set of parties $S = S_0 \cup S_1$ with mixed sides (left shares from S_0 and right shares from S_1) into an identity key (L_j, R_j), where L_j represents the aggregation over S_0 and R_j the aggregation over S_1. Note that the aggregation is computed in the exponent.

Decryption. The identity key holder can decrypt a ciphertext (c_0, c_1) for an index j and idenitity ID using the j'th component of the identity key (L_j, R_j), by computing $e(c_0, L_j) \cdot e(c_1, R_j)$. Two-sided correctness follows from the decryption equation:

$$
\begin{aligned}
&e(c_0, L_j) \cdot e(c_1, R_j) \\
&= e(Y_j^r, \mathcal{H}(j, \mathsf{ID})^{z_j \sum_i^{S_0} l_{i,S} \cdot s_i^{(j)}}) \cdot e(Z_j^r, \mathcal{H}(j, \mathsf{ID})^{y_j \sum_i^{S_1} l_{i,S} \cdot s_i^{(j)}}) \\
&= e(g_1, \mathcal{H}(j, \mathsf{ID}))^{r y_j z_j \sum_i^{S} l_{i,S} \cdot s_i^{(j)}} = e(g_1, \mathcal{H}(j, \mathsf{ID}))^{r y_j z_j \alpha_j} = k
\end{aligned}
$$

Security of BTIB−KEM−1. In addition to two-sided correctness, we need to prove semantic security and one-sided security. Note that the main challenge in the proof over [15] is that we must simulate the identity key derivation oracle O^{DIdk} for semantic security as we work in the identity-based setting.

Lemma 5.1 (IND−SID−CPA security of BTIB−KEM−1). *The* BTIB−KEM−1 *construction is* IND−SID−CPA *secure (Definition 4.3)*

if the BDDH assumption (Definition C.4) holds in the underlying pairing ensemble.

Lemma 5.2 (One-sided security of BTIB−KEM−1). *The* BTIB−KEM−1 *construction is one-sided secure (Definition 4.4) if the XDH assumption (Definition C.5) holds in the underlying pairing ensemble.*

The proofs are deferred to Appendix F.1 and F.2 respectively.

5.2 BTIB−KEM−2 (Constant-Size Public Key)

For simplicity, we restrict ourselves to the non-threshold case in this presentation, but we note that it can be thresholdized using the same techniques as in our first construction. We give a formal construction, along with formal security proofs, in Appendix H.

Our construction is based on the Boneh-Boyen 2-layer IBE construction [7]. In a 2-layer (or hierarchical) IBE [40], there are two layers of identities (e.g., (I_1, I_2)). With the master key, one can derive the identity key for any first-layer identity I_1. The first-layer identity key can then be used to derive the identity key for any layer-two identity that has I_1 as prefix, i.e., $(I_1, *)$, but not $(I_1', *)$ for $I_1 \neq I_1'$. Hence, the layers of identities act as hierarchical domains.

Inspired by [15, 50], we use 2-layers in order to compress all public keys for the ℓ different positions in one layer. That is, instead of encrypting under pk_j to ID, we encrypt under a global pk and to identity (j, ID). . As secret keys, we distribute the identity keys for the positions $j \in [\ell]$. Hence, one can derive an identity key for ID at position j (that is, the identity key for (j, ID)) using the j'th secret key. Note that the encryptor needs to know only the master public key pk, which is independent of position j.

5.2.1 Boneh-Boyen IBE [7]. We recall the non-threshold Boneh-Boyen IBE construction for two layers. For threshold version, see [9].

Setup and key generation. Choose random $\alpha, b \leftarrow\!\!\$\ \mathbb{Z}_p$ and $u_1, u_2 \leftarrow\!\!\$\ \mathbb{G}_1$. Define functions $L_1, L_2 \colon \mathbb{Z}_p \to \mathbb{G}_1$ such that $L_1(x) = g_1^{\alpha x} \cdot u_1$ and $L_2(x) = g_1^{\alpha x} \cdot u_2$. Publish $g_1^{\alpha}, g_2^{b}, u_1$, and u_2 as public parameters. The master secret key is $g_1^{\alpha b}$.

Identity key derivation. Given an identity I_1 on the first layer , sample $r_1 \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $\mathsf{idk}^{I_1} = (g_1^{\alpha b} \cdot L_1(I_1)^{r_1}, g_2^{r_1})$. And for the second layer, given $ID = (I_1, I_2)$, sample $r_2 \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $\mathsf{idk}^{\mathsf{ID}} = (\mathsf{idk}_1^{I_1} \cdot L_2(I_2)^{r_2}, \mathsf{idk}_2^{I_1}, g_2^{r_2})$.

Encryption. To encrypt to identity (I_1, I_2), choose a random $s \in \mathbb{Z}_p$, set $k = e(g_1^{\alpha}, g_2^{b})^s$ and $ct = (g_2^s, L_1(I_1)^s, L_2(I_2)^s)$.

Decryption. Given idk^{ID} and ct that is encrypted under ID, compute k as $k \leftarrow e(\mathsf{idk}_1^{\mathsf{ID}}, ct_1)/(e(ct_2, \mathsf{idk}_2^{\mathsf{ID}}) \cdot e(ct_3, \mathsf{idk}_3^{\mathsf{ID}}))$.

5.2.2 Our BTIB−KEM−2 *Construction.* In our construction, we use two instances of 2-layer Boneh-Boyen IBE (one per side) with modifications to account for our new correctness and security notions, as follows.

Setup and Key Generation. The Setup algorithm generates an asymmetric pairing ensemble E. The key space is $\mathcal{K}(\lambda) = \mathbb{G}_T$. The identity space in our second construction is $\mathsf{IDSpace} = \mathbb{Z}_p$, but can be arbitrarily extended using a hash function to $\mathbb{Z}_p$. We also use a cryptographic hash function $\mathcal{H}$ with output domain $\mathbb{G}_1$. Similarly to our first construction, we choose random $\alpha, y, z \in \mathbb{Z}_p$. As we consider two sides, we use an instance for the left side in which we set the master key for position j to be $\mathcal{H}(j)^{\alpha z}$ and for the right side $\mathcal{H}(j)^{\alpha y}$. That is, from the notation above, g_1^b is $g_1^{w_j z}$ where $\mathcal{H}(j) = g_1^{w_j}$. Denote the parameters used in the first instance by u_1, u_2, L_1, L_2 and for the second instance by v_1, v_2, R_1, R_2 (i.e., representation of the functions). The public key is set to $(g_1^{\alpha}, g_2^{y}, g_2^{z}, g_2^{\alpha yz}, u_1, u_2, v_1, v_2)$, which is independent of ℓ.

Secret keys. In our construction, the secret keys for each position $j \in [\ell]$ conceptually corresponds to an identity key for identity j. Hence, we generate the secret keys similar to how one would generate first-layer identity keys in the Boneh-Boyen IBE from the master secret key, except that we extend the secret key by two elements (the second and the fourth), which are needed for our two-sided correctness. That is, for every j, we first sample $r_{1,0}, r_{1,1} \leftarrow\!\!\$\ \mathbb{Z}_p$ and set left secret key for position j as

$$\mathsf{sk}_0^{(j)} = (\mathcal{H}(j)^{\alpha z} \cdot L_1(j)^{r_{1,0}}, g_2^{y r_{1,0}}, g_2^{r_{1,0}}, g_2^{\alpha z}),$$

and the right key as $\mathsf{sk}_1^{(j)} = (\mathcal{H}(j)^{\alpha y} \cdot R_1(j)^{r_{1,1}}, g_2^{z r_{1,1}}, g_2^{r_{1,1}}, g_2^{\alpha y})$.

Key derivation. Again, we follow the key derivation in Boneh-Boyen scheme, except with new elements similar to the first layer. For a given identity ID, we sample $r_{2,0}, r_{2,1} \leftarrow\!\!\$\ \mathbb{Z}_p$ and set the left identity key for position j as

$$\mathsf{idk}_{\mathsf{ID},0}^{(j)} = (\mathsf{sk}_{0,1}^{(j)} \cdot L_2(\mathsf{ID})^{r_{2,0}}, \mathsf{sk}_{0,2}^{(j)}, g_2^{y r_{2,0}}, \mathsf{sk}_{0,3}^{(j)}, g_2^{r_{2,0}}, \mathsf{sk}_{0,4}^{(j)}),$$

and $\mathsf{idk}_{\mathsf{ID},1}^{(j)} = (\mathsf{sk}_{1,1}^{(j)} \cdot R_2(\mathsf{ID})^{r_{2,1}}, \mathsf{sk}_{1,2}^{(j)}, g_2^{z r_{2,1}}, \mathsf{sk}_{1,3}^{(j)}, g_2^{r_{2,1}}, \mathsf{sk}_{1,4}^{(j)})$.

Encryption. To account for two-sided correctness, similar to our first construction, we modify the ciphertext such that we replace g_2^s by g_2^{ys} in c_0 and by g_2^{zs} in c_1. Furthermore, in this case we need to mask these values with random elements t_0 and t_1, respectively, in order to account for one-sided security. This modification entails additional elements in the ciphertext that are needed to cancel the masks t_0 and t_1 during decryption. That is, in encryption we sample $s, t_0, t_1 \leftarrow\!\!\$\ \mathbb{Z}_p$, and set the key $k = e(\mathcal{H}(j)^s, g_2^{\alpha yz})$. The resulting ciphertext is $c = (c_0, c_1)$, where

$$c_0 = (g_2^{ys+t_0}, L_1(j)^s, L_2(\mathsf{ID})^s, L_1(j)^{t_0}, L_2(\mathsf{ID})^{t_0}, \mathcal{H}(j)^{t_0}),$$

and $c_1 = (g_2^{zs+t_1}, R_1(j)^s, R_2(\mathsf{ID})^s, R_1(j)^{t_1}, R_2(\mathsf{ID})^{t_1}, \mathcal{H}(j)^{t_1})$. Note that the last three elements on each side are necessary to cancel the masks t_0 and t_1 during decryption.

Decryption. The decryption follows from a similar formula as Boneh-Boyen, but in the bipartite setting: Let $\mathsf{idk}_0, \mathsf{idk}_1$ be $\mathsf{idk}_{\mathsf{ID},0}^{(j)}$, $\mathsf{idk}_{\mathsf{ID},1}^{(j)}$, respectively. We compute k as

$$k \leftarrow \frac{e(\mathsf{idk}_{b,1}, c_{b,1})}{\prod_{i=2}^{6} e(c_{b,i}, \mathsf{idk}_{b,i})}$$

for $b \in \{0, 1\}$. It can be computed by the left or right key since, in this simplified version, the threshold is 1. In the thresholdized version, the final key is computed as the multiplication of two decryption outputs, similar to our first construction in Figure 8.

Correctness and Security. The two-sided correctness immediately follows the decryption formula when applying the same thresholding technique as in the first construction.

Intuitively, the semantic security follows from the semantic security of each one of 2-layer IBE instances. While we use ideas from [9], due to our changes some challenges need to be resolved.

The one-sided security relies on the masks t_0 and t_1 that we applied to the ciphertexts. Note that without masking, the one-sided security is broken by, e.g., the check that $e(R_1(j)^s, g_2^y) = e(R_1(j), g_2^{ys})$. Intuitively, masking prevents this attack since the adversary does not know $R_1(j)^{t_0}$ or $L_1(j)^{t_1}$. Also, we had to introduce the random oracle in the master secret keys ($\mathcal{H}(j)^{\alpha z}$ and $\mathcal{H}(j)^{\alpha y}$) in order to simulate all secret keys but those for position j^* on side $1-d$ in the one-sided security proof. Our proofs rely on assumptions that can be proven secure in the Generic Group Model. The full proofs appear in Appendix H.

6 Efficiency Analysis

We next discuss the efficiency of our constructions with respect to the ciphertext and key size, as well as computational efficiency.

Parameter Size. The table in Figure 9 summarizes the parameter sizes in the resulting CCA-secure TT−KEM construction that is obtained by plugging both of our BTIB−KEM construction into the transformation. Observe that both constructions are relatively

Param	With BTIB−KEM−1	With BTIB−KEM−2
$\lvert c\rvert$	$2\mathbb{G}_1 + \lvert\mathsf{vk}\rvert + \lvert\sigma\rvert + \log(\ell)$	$10\mathbb{G}_1 + 2\mathbb{G}_2 +$ $\lvert\mathsf{vk}\rvert + \lvert\sigma\rvert + \log(\ell)$
$\lvert\mathsf{pk}\rvert$	$3\ell\mathbb{G}_1$	$5\mathbb{G}_1 + 3\mathbb{G}_2$
$\lvert\mathsf{sk}_i\rvert$	$2\ell\mathbb{Z}_p$	$\ell(\mathbb{G}_1 + 3\mathbb{G}_2)$
$\lvert d_i\rvert$	$1\mathbb{G}_2$	$1\mathbb{G}_1 + 5\mathbb{G}_2$
$\lvert d\rvert$	$2\mathbb{G}_2$	$2(1\mathbb{G}_1 + 5\mathbb{G}_2)$

Figure 9: Summary of the parameter sizes of CCA-secure TT−KEM, when instantiating with BTIB−KEM−1 or BTIB−KEM−2 and a signature scheme Σ as building blocks. $\lvert\mathsf{vk}\rvert$ and $\lvert\sigma\rvert$ refer to the length of the verification key and signature of Σ.

size-efficient in the important parameters[9] that is the ciphertext c, the public key pk and the partial decryptions d_i. Also, observe that it is possible to combine at least t partial decryptions $\{d_i\}_{i\in S}$ from a set of parties S into a single decryption key d to save even more space. This decryption key d is only twice the size of any individual d_i. This works because both our constructions support the pre-aggregation of identity key shares through CombIdk.

In the TT−KEM construction, the number of positions ℓ is determined according to the codeword size in the underlying fingerprinting code. When we use the fingerprinting code due to Tardos [49], the size of ℓ is $O(n^2 \log^2 n)$.

Both of our constructions are concretely computationally efficient, with execution times in the order of milliseconds. An exact overview is laid out in Appendix I.

[9]We consider them most important since they are the most likely to be stored on the blockchain when using our protocol in scenarios such as mempool privacy.

7 Consistency

In this section, we discuss the consistency property in TT−KEM.

Recall that the consistency requirement ensures that any coalition of parties will get the same decryption result, even with adversarially generated ciphertext and decryption shares. Our consistency notion is very strong, in the sense that it even holds against adversaries that corrupt *all* shareholders. On the other hand, we can guarantee consistency only against adversaries who do not know the tracing key tk. Intuitively, this is because we want to ensure through a trapdoored proof that the ciphertext is *well-formed* while still allowing tracing, where the tracer sends queries to the decoder box D with ciphertexts that have mismatching left and right parts. [10] We present a formal consistency definition in Appendix J.1.

Construction from Disjunctive NIZK-PoKs. In order to achieve consistency for our CCA-secure TT−KEM construction from Section 4.2, we need to add a procedure for verification of the partial decryption shares and to attach a well-formedness proof of the ciphertext with a trapdoor for the tracer. For simplicity, we present the modifications needed, when starting from the concrete BTIB−KEM construction in Section 5.1. The ideas presented here can be generalized to any BTIB−KEM construction. Since the validity of partial decryptions can be easily verified using the BLS equation (see Appendix J.1.1 for details), we focus on the well-formedness proof.

Well-formedness of Ciphertext via a Trapdoor. In KGen, we generate a discrete-log tuple (q, Q), where $q \leftarrow\!\$\ \mathbb{Z}_p$ and $Q \leftarrow g_1^q$. In the transformation in Section 4.2, the trapdoor q is included in the tracing key while the challenge Q is included in the public key. In Enc, the encryptor builds a NIZK proof of knowledge for the following relation R. For a statement $\chi = (\mathsf{pk}, (c_0, c_1, j, \mathsf{vk}_e))$ and a witness ω, we say that $(\chi, \omega) \in R$ if and only if it holds that:

$$(c_0 = Y_j^\omega \wedge c_1 = Z_j^\omega) \vee (Q = g_1^\omega).$$

Hence, the encryptor needs to prove that he knows a witness ω such that $(\chi, \omega) \in R$, where χ is extracted from its ciphertext. Notice that in normal encryption, the encryptor generates a valid proof by using the randomness r that it used to encrypt c_0 and c_1 as witness. In tracing, on the other hand, the tracer can generate a valid proof by proving that it knows the trapdoor.

As discussed, this proof does not break traceability of TT−KEM. Indeed, given a valid proof, the adversary cannot distinguish if it was generated via normal encryption or during tracing. This holds due to the zero-knowledge property of the proof system. Finally, the resulting scheme satisfies the consistency notion, due to the knowledge-soundness of the proof system as well as the verifiability of the partial decryption shares. For more details, see Appendix J.2.

We also highlight that the proof system can be efficiently instantiated using disjunctive Schnorr proofs of knowledge, which is a standard technique [27]. The additional proof would add three $\mathbb{G}_1$ elements and four $\mathbb{Z}_p$ elements to the ciphertext over the result for BTIB−KEM−1 in Figure 9.

[10]Note that according to [51], consistency with public tracing (i.e., with no tracing key) is hard. Thus, relying on the tracing key to achieve consistency is reasonable.

Acknowledgements

This work is supported in part by EPSRC grant EP/Y001680/1, by the European Research Council (ERC) under the European Union's Horizon 2020 and Horizon Europe research and innovation programs (grant CRYPTOLAYER-101044770), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts, through their joint sponsorship of the National Research Center for Applied Cybersecurity ATHENE, and by the German Research Foundation (DFG) via the DFG CRC 1119 CROSSING project S7.

References

[1] M. Abdalla, A. W. Dent, J. Malone-Lee, G. Neven, D. H. Phan, and N. P. Smart. Identity-based traitor tracing. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 361–376. Springer, 2007.

[2] A. Agarwal, R. Fernando, and B. Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. Cryptology ePrint Archive, Paper 2024/1575, 2024.

[3] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Paper 2005/385, 2005.

[4] L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Paper 2005/417, 2005.

[5] B. Barak. An intensive introduction to cryptography: Chosen ciphertext security. https://intensecrypto.org/public/lec_06_CCA.html, 2023. Accessed: 2025-01-09.

[6] A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 470–491. Springer, 2015.

[7] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[8] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 440–456. Springer, 2005.

[9] D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2006.

[10] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[11] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[12] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[13] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In C. Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[14] D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In P. Ning, P. F. Syverson, and S. Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 501–510. ACM, 2008.

[15] D. Boneh, A. Partap, and L. Rotem. Accountability for misbehavior in threshold decryption via threshold traitor tracing. In *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 317–351. Springer, 2024.

[16] D. Boneh, A. Partap, and L. Rotem. Traceable secret sharing: Strong security and efficient constructions. In *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part V*, volume 14924 of *Lecture Notes in Computer Science*, pages 221–256. Springer, 2024.

[17] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In V. Atluri, B. Pfitzmann, and P. McDaniel, editors, *ACM CCS 2004: 11th Conference on Computer and Communications Security*, pages 168–177, Washington, DC, USA, Oct. 25–29, 2004. ACM Press.

[18] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 452–465, Santa Barbara, CA, USA, Aug. 27–31, 1995.

[19] J. Bormet, S. Faust, H. Othman, and Z. Qu. BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. Cryptology ePrint Archive, Paper 2024/1533, 2024.

[20] 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.

[21] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 501–510, 2010.

[22] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 255–271. Springer, 2003.

[23] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 255–271. Springer, 2003.

[24] J. Chen, H. W. Lim, S. Ling, H. Wang, and H. Wee. Shorter IBE and signatures via asymmetric pairings. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, volume 7708 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2012.

[25] A. R. Choudhuri, S. Garg, J. Piet, and G.-V. Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. Cryptology ePrint Archive, Paper 2024/669, 2024.

[26] A. R. Choudhuri, S. Garg, G.-V. Policharla, and M. Wang. Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive, Paper 2024/1516, 2024.

[27] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994.

[28] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[29] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges, 2019.

[30] Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 100–115. Springer, 2003.

[31] S. Dziembowski, S. Faust, T. Lizurej, and M. Mielniczuk. Secret sharing with snitching. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 840–853. ACM, 2024.

[32] S. Dziembowski, S. Faust, and J. Luhn. Shutter network: Private transactions from threshold cryptography. *Cryptology ePrint Archive*, 2024.

[33] S. Eskandari, S. Moosavi, and J. Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 170–189. Springer, 2020.

[34] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[35] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptol.*, 26(1):80–101, 2013.

[36] D. Galindo and I. Hasuo. Security notions for identity based encryption. *Cryptology ePrint Archive*, 2005.

[37] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In Y. Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

[38] V. Goyal, Y. Song, and A. Srinivasan. Traceable secret sharing and applications. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part*

III, volume 12827 of *Lecture Notes in Computer Science*, pages 718–747. Springer, 2021.

[39] F. Guo, Y. Mu, and W. Susilo. Identity-based traitor tracing with short private key and short ciphertext. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, volume 7459 of *Lecture Notes in Computer Science*, pages 609–626. Springer, 2012.

[40] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.

[41] A. Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptol.*, 17(4):263–276, 2004.

[42] C. S. Jutla and A. Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013.

[43] S. Maitra and D. J. Wu. Traceable prfs: Full collusion resistance and active security. In *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 439–469. Springer, 2022.

[44] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In H. Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990.

[45] D. H. Phan and V. C. Trinh. Identity-based trace and revoke schemes. In *Provable Security - 5th International Conference, ProvSec 2011, Xi'an, China, October 16-18, 2011. Proceedings*, volume 6980 of *Lecture Notes in Computer Science*, pages 204–221. Springer, 2011.

[46] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[47] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology—EUROCRYPT'97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings 16*, pages 256–266. Springer, 1997.

[48] Shutter Network. Introducing Shutter – Combating Front Running and Malicious MEV Using Threshold Cryptography. https://blog.shutter.network/introducing-shutter-network-combating-frontrunning-and-malicious-mev-using-threshold-cryptography/, 2021. Accessed: 2025-01-05.

[49] G. Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2):10:1–10:24, 2008.

[50] M. Zhandry. New techniques for traitor tracing: Size $n^{1/3}$ and more from pairings. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 652–682. Springer, 2020.

[51] M. Zhandry. White box traitor tracing. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 303–333. Springer, 2021.

[52] L. Zhou, K. Qin, A. Cully, B. Livshits, and A. Gervais. On the just-in-time discovery of profit-generating transactions in DeFi protocols. pages 919–936. IEEE Computer Society Press, 2021.

[53] L. Zhou, K. Qin, C. Ferreira Torres, D. V. Le, and A. Gervais. High-frequency trading on decentralized on-chain exchanges. pages 428–445. IEEE Computer Society Press, 2021.

A CCA Attacks on BPR24 Constructions

We briefly discuss the constructions of [15], and present a CCA attack against them. We stress again that the CCA attack is practically relevant to applications such as mempool privacy, since the committee that decrypts transactions basically acts as a decryption oracle that returns partial decryptions. The CCA attack is on the constructions by [15] is the standard attack on homomorphic encryption schemes. For their second construction, the homomorphism is broken by hashing the result of the decryption and using that as key. The attack, however, can still be carried out given the partial decryptions.

First Construction. The first construction of [15] is reminiscent of ElGamal encryption. The j^{th} public key is a triple $X_j = g^{x_j}, Y_j = g^{y_j}, Z_j = g^{z_j}$. The left and right ciphertexts are Y_j^r and Z_j^r respectively for a randomly sampled r, and the key k is X_j^r. The CCA adversary, on input the challenge ciphertext (Y_j^r, Z_j^r), asks for decryption of $(\hat{C}_l, \hat{C}_r) = \left(\left(Y_j^r\right)^2, \left(Z_j^r\right)^2\right)$. The corresponding key would be $\hat{k} = X_j^{2r}$ (can be derived from decryption shares received from the decryption oracle). Now, the the actual key k can be computed by computing $\hat{k}^{1/2}$.

Second Construction. In the second construction, the key is derived by evaluating a hash function H_2 on the result W of the combine-partial-decryptions. However, the adversary still learns the partial decryptions, hence it can derive W which exhibits homomorphic structure. More concretely, in their second construction, each component of the ciphertext is a pair of group elements $C_\ell = (u_\ell, v_\ell) \in \mathbb{G}_1 \times \mathbb{G}_2$, and $C_r = (u_r, v_r) \in \mathbb{G}_1 \times \mathbb{G}_2$. Moreover, for both $b \in \{\ell, r\}$, if $d_{(i,b)}$ is the output of the partial decryption for the i^{th} share of C_b, then it holds that $d^2_{(i,b)}$ is the output of the partial decryption for the i^{th} share of $C_b^2 = (u_b^2, v_b^2)$.

Thus, given the challenge ciphertext (C_ℓ, C_r) the adversary can submit (C_ℓ^2, C_r^2) for the partial decryption. As the submitted ciphertext is different from the challenge ciphertext, the adversary can make the query for t(threshold) number of shares, and get d_i^2 and thus derive d_i for t many parties. The adversary combines t many d_is to compute W, and therefore derive $K = H_2(W)$.

B More related work

There has been extensive research on traitor tracing in the broadcast encryption setting (see, e.g., [15, 50] for a good overview). Traitor tracing in the broadcast encryption setting is mostly motivated by trying to catch people who illegally sell or replicate their decryption device for, say, pay TV broadcast. In this setting [30] is the first work to consider CCA-secure traitor tracing, using Cramer-Shoup transformation [28]. In particular, they show a k-resilient construction with ciphertext overhead of $O(k)$, which is less efficient than our construction, where we achieve a constant-size ciphertext with full-resiliency. Other works [1, 39, 45] studied traitor tracing IBE in the broadcast encryption setting. Specifically, they extend the traitor tracing in broadcast encryption to the identity-based setting, such that each user gets a decryption key for some identity group. Hence, the decoder box that is constructed by a set of users can decrypt only for their corresponding identity groups (as users cannot derive new identity keys). Our model (and that of [15]) differs from all these works, as we aim to trace colluding master-key shareholders of thresholdized committees, while traitor tracing for broadcast encryption rather focuses on tracing end-users that construct illegal decryption devices in the non-threshold setting.

In [43], they construct a traceable PRF and use it as building block for constructing CCA-secure fingerprinting-based traitor tracing in the symmetric key setting, where in this paper we consider the public key setting. Consistency, which is another property that we

study, was defined and studied for the non-threshold setting in [51]. In their paper they suggest to work in the white-box decoder model, in which the tracer can access also the code of the decoder instead of only black-box access.

C Additional Definitions

C.1 Cryptographic Building Blocks

Definition C.1 (One-Time Signature Schemes). A one-time signature scheme Σ is a tuple of algorithms $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ such that for all messages m and all $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ it holds that

$$\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1.$$

Further, we require Σ to achieve EUF−OT−CMA-security.

$\mathsf{Game{-}EUF{-}OT{-}CMA}^{\Sigma}_{\mathcal{A}}(1^\lambda)$	$O^{\mathrm{Sign}}(m)$
$M \leftarrow \emptyset$	**if** $q = 0$ **then return** $\perp$
$q \leftarrow 1$	$q \leftarrow q - 1$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$	$M \leftarrow M \cup \{m\}$
$(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{O^{\mathrm{Sign}}}(1^\lambda, \mathsf{vk})$	$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, m)$
if $m^* \in M$ **then return** 0	**return** σ
return $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$	

Figure 10: EUF−OT−CMA **security game for signature schemes.**

Definition C.2 (Existential Unforgeability under Chosen Message Attacks for One-Time Signatures (EUF−OT−CMA)*).* We say that a one-time signature scheme Σ is EUF−OT−CMA-secure if for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathsf{Game{-}EUF{-}OT{-}CMA}^{\Sigma}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where Game−EUF−OT−CMA is as defined in Figure 10.

Definition C.3 (Non-Interactive Zero-Knowledge Proofs of Knowledge). Let R be an NP-relation and of statement-witness pairs (χ, ω). A non-interactive zero-knowledge proof of knowledge (NIZK-PoK) system Π for relation R is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following properties:

- Correctness: For all $(\chi, \omega) \in \mathsf{R}$ it holds that
$$\mathsf{Verify}(\chi, \mathsf{Prove}(\chi, \omega)) = 1.$$
- Knowledge-soundness: There exists a PPT extraction algorithm Ext such that for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl with
$$\Pr\left[\begin{array}{c|c} (\chi, \omega) \notin \mathsf{R} \wedge & (\chi, \pi) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda) \\ \mathsf{Verify}(\chi, \pi) = 1 & \omega \leftarrow\!\!\$\ \mathsf{Ext}^{\mathcal{A}(\cdot)}(\chi, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda).$$
- Zero-knowledge: There exists a PPT simulator Sim such that for all $(\chi, \omega) \in \mathsf{R}$ it holds that
$$(\mathsf{CRS}, \chi, \mathsf{Prove}(\chi, \omega)) \approx_c (\mathsf{CRS}, \chi, \mathsf{Sim}(\chi)).$$

All algorithms receive a common CRS $\leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ as implicit input.

C.2 Well-Established Bilinear Pairing Assumptions

The Bilinear (Decisional) Diffie-Hellman (BDDH) assumption was introduced in [12] and since been the bedrock of Identity-based cryptography and its extensions, and, in general, pairing-based cryptography ([23, 37, 41, 46] and many more).

Definition C.4 (Bilinear Decisional Diffie-Hellman). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the bilinear decisional Diffie-Hellman (BDDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries $\mathcal{B}$, it holds that

$$\left|\Pr\left[\mathcal{B}(E, \vec{x}, e(g_1, g_2)^{abc}) = 1\right] - \Pr\left[\mathcal{B}(E, \vec{x}, e(g_1, g_2)^{v}) = 1\right]\right|$$

is negligible in λ where $E \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$, $a, b, c, v \leftarrow\!\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \left(g_1, g_1^a, g_1^b, g_2, g_2^a, g_2^c\right).$$

The XDH assumption was first discussed in [10], and formalized in [4]. Since then, it has been used in numerous constructions including Identity-Based Encryption [24], Inner-Product Encryption [6], Group Signatures [3], Leakage-Resilient Encryption [21], and Non-Interactive Zero-knowledge proofs [42].

Definition C.5 (XDH Assumption). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the external Diffie-Hellman (XDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries $\hat{\mathcal{B}}$ it holds that

$$\left|\Pr\left[\mathcal{B}(E, \vec{x}, g_1^{ab}) = 1\right] - \Pr\left[\mathcal{B}(E, \vec{x}, g_1^{v}) = 1\right]\right|$$

is negligible in λ where $E \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$, and $a, b, v \leftarrow\!\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \left(g_1, g_1^a, g_1^b, g_2\right).$$

C.3 Traceable Threshold KEM

Definition C.6 (Correctness of TT−KEM*).* A TT−KEM protocol Π is correct if for all $\lambda \in \mathbb{N}$, all polynomials $\varepsilon(\lambda)$, all $n \geq t > 0$, all $S \subseteq [n]$ with $|S| \geq t$ it holds that

$$\Pr\left[\begin{array}{c|l} k = k' \wedge & (\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{tk}) \\ \mathsf{Verify}(c, \pi) = 1 & \quad \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon(\lambda)}) \\ & (k, c, \pi) \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}) \\ & \forall i \in S\colon d_i \leftarrow \mathsf{PDec}(\mathsf{sk}_i, c) \\ & k' \leftarrow \mathsf{Comb}(S, \{d_i\}_{i\in S}, c) \end{array}\right] = 1.$$

Further, we require that the keys generated by Enc are uniformly distributed over the key-space $\mathcal{K}(\lambda)$.

The most important detail about IND−CCA security of TT−KEM is that the adversary gets access to the tracing key. This reflects the fact that the tracer is not trusted for security. Even if the tracer is corrupt, full CCA security holds as long as less than t shareholders are corrupt.

*Definition C.7 (*IND−CCA*-security of* TT−KEM*).* A TT−KEM protocol π is IND−CCA-secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function negl such that

$$\Pr\left[\mathsf{Game{-}IND{-}CCA}^{\mathsf{TT-KEM}}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where $\mathsf{Game{-}IND{-}CCA}^{\mathsf{TT-KEM}}_{\mathcal{A}}$ is defined in Figure 11.

$\underline{\mathsf{Game\text{–}IND\text{–}CCA}^{\mathsf{TT-KEM}}_{\mathcal{A}}(1^\lambda)}$

$\mathsf{ctr} \leftarrow 0; b \leftarrow\$ \{0,1\}$

$(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{tk})$

$\quad \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon(\lambda)})$

$(C, \mathsf{st}) \leftarrow\$ \mathcal{A}(\mathsf{st}_1, \mathsf{pk}, \mathsf{tk})$

if $C \not\subseteq [n] \vee |C| \geq t$ **then**

  return 0

$k_0 \leftarrow\$ \mathcal{K}(\lambda)$

$(k_1, c^*, \pi^*) \leftarrow\$ \mathsf{Enc}(\mathsf{pk})$

$\mathsf{chal} \leftarrow ((c^*, \pi^*), k_b), \{\mathsf{sk}_i\}_C)$

$b' \leftarrow\$ \mathcal{A}_3^{O^{\mathsf{PDec}}}(\mathsf{st}, \mathsf{chal})$

return $b \stackrel{?}{=} b'$

$\underline{O^{\mathsf{PDec}}((c,\pi), i)}$

if $\mathsf{Verify}(c, \pi) = 0$ **then**

  return $\perp$

if $c = c^*$ **then**

  $\mathsf{ctr} \leftarrow\$ \mathsf{ctr} + 1$

  if $\mathsf{ctr} \geq t - |C|$ **then**

    return $\perp$

$d_i \leftarrow \mathsf{PDec}(c, \mathsf{sk}_{i,0})$

return d_i

Figure 11: IND–CCA **security game for** TT–KEM **and standard threshold KEM.**

$\underline{\mathsf{Game\text{–}Trace}_{\mathcal{A},\varepsilon}(1^\lambda)}$

$(n, t, \mathsf{st}_1) \leftarrow\$ \mathcal{A}_1(1^\lambda)$

$(\mathsf{pk}, \{\mathsf{sk}_i\}_{[n]}, \mathsf{tk}) \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon(\lambda)})$

$(\mathcal{J}, \mathsf{st}_2) \leftarrow\$ \mathcal{A}_2(\mathsf{st}_1, \mathsf{pk})$

if $\mathcal{J} \not\subseteq [n]$ **then** abort

$D \leftarrow\$ \mathcal{A}_3(\mathsf{st}_2, \{\mathsf{sk}_i\}_{i\in C})$

$\mathcal{J}' \leftarrow\$ \mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, 1^{1/\varepsilon(\lambda)})$

return $(\mathsf{pk}, D, \mathcal{J}, \mathcal{J}')$

Figure 12: Traceability game for TT–KEM**.**

Definition C.8 ((ε,δ)-Traceability of TT–KEM*).* Let $\varepsilon(\lambda)$ and $\delta(\lambda)$ be functions in the security parameter. A TT–KEM scheme is (ε,δ)-traceable, if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ and for all $\lambda \in \mathbb{N}$ the following conditions hold

$$\Pr[\mathsf{GoodTr}] \geq \Pr[\mathsf{GoodDec}] - \delta(\lambda) \quad \text{and} \quad \Pr[\mathsf{BadTr}] \leq \delta(\lambda),$$

where $(\mathsf{pk}, D, \mathcal{J}, \mathcal{J}') \leftarrow\$ \mathsf{Game\text{–}Trace}_{\mathcal{A},\varepsilon}(1^\lambda)$ (Figure 12) and the three events GoodTr, BadTr and GoodDec are defined as follows:

- GoodTr holds when $\mathcal{J}' \neq \emptyset$ and $\mathcal{J}' \subseteq \mathcal{J}$, which indicates that at least one real traitor has been traced and no non-traitors were framed by the tracing algorithm.
- BadTr holds when at least one non-traitor was framed by the tracing algorithm ($\mathcal{J}' \neq \emptyset$ and $\mathcal{J}' \not\subseteq \mathcal{J}$).
- GoodDec indicates that the decoder is good for pk. More formally, when $P(D) \geq 1/2 + \varepsilon(\lambda)$ where

$$P(D) \coloneqq \Pr\left[D(c, k_b) = b : \begin{array}{c}((c,\tau), k_0) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}) \\ k_1 \leftarrow\$ \mathcal{K}_\lambda, b \leftarrow\$ \{0,1\}\end{array}\right].$$

Definition C.9 (Traceability of TT–KEM*).* A TT–KEM scheme is traceable if there exists a negligible function ν such that TT–KEM is $(1/p, \nu)$-traceable for every positive polynomial $p(\lambda)$.

D TTIB-KEM From BTIB-KEM

In this section, we extend the TT–KEM definition to the identity-based setting, providing the definition of traceable threshold identity-based KEM (TTIB–KEM). We then discuss a transformation from our main building block BTIB–KEM to TTIB–KEM.

D.1 Traceable Threshold Identity-Based KEM

We next present our definition of TTIB–KEM, which extends the standard identity-based KEM by a tracing functionality.

We model traceability for threshold identity-based KEM schemes by lifting the definition from [15] to the identity-based setting. Our model accounts for decoders that can decrypt ciphertexts of a chosen subset of identities (the set $\mathcal{I}^*$ in Figure 13), that is determined in the beginning of the scheme.

Definition D.1. A traceable threshold identity-based KEM scheme (TTIB–KEM) is a tuple of protocols (KGen, Enc, DIdk, CombIdk, Dec, Trace) with KEM key-space $\mathcal{K}(\lambda)$ and identity-space IDSpace.

- $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon(\lambda)})$. The KGen algorithm receives the security parameter 1^λ, the number of parties n, the threshold t where $n \geq t > 0$ and the tracing parameter ε, which is a lower bound on the distinguishing advantage of decoders that must be traceable by this scheme. It outputs a public key pk, the secret key shares $\mathsf{sk} = \{\mathsf{sk}_i\}_{i\in[n]}$, and a tracing key tk.
- $(k, c) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{ID})$. Given a public key pk and an identity $\mathsf{ID} \in \mathsf{IDSpace}$, the encryption algorithm returns a key $k \in \mathcal{K}(\lambda)$ and a ciphertext c.
- $\mathsf{idk}_i \leftarrow \mathsf{DIdk}(\mathsf{sk}_i, \mathsf{ID})$. The identity key derivation algorithm DIdk receives the secret key share of party i and an identity ID. It returns a partial identity key on ID as idk_i.
- $\mathsf{idk} \leftarrow \mathsf{CombIdk}(S, \{\mathsf{idk}_i\}_{i\in S})$. The CombIdk algorithm aggregates partial identity keys from a set S of size at least t and outputs the resulting identity key idk.
- $k/\perp \leftarrow \mathsf{Dec}(\mathsf{idk}, c)$. The decryption algorithm receives an identity key idk and a ciphertext c. It returns the resulting key k or $\perp$ upon failure.
- $\mathcal{J} \leftarrow\$ \mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, \mathcal{I}, 1^{1/\varepsilon(\lambda)})$. The Trace algorithm is given the public key pk, the tracing key tk, a set of possible identities $\mathcal{I}$, the parameter ε, and black-box access to the decoder D. It outputs a set of traitors $\mathcal{J} \subseteq [n]$.

We omit the correctness and IND–SID–CPA definitions as they are standard and not affected by traceability.

Definition D.2. [(ε,δ)-Traceability of TTIB–KEM] For functions in the security parameter $\varepsilon(\lambda)$ and $\delta(\lambda)$, a TTIB–KEM scheme is (ε,δ)-traceable, if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ and for all $\lambda \in \mathbb{N}$ the following conditions hold

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

with $(\mathsf{pk}, \mathcal{I}^*, D, \mathcal{J}, \mathcal{J}') \leftarrow\$ \mathsf{Game\text{–}Trace\text{–}ID}_{\mathcal{A},\varepsilon}(1^\lambda)$ (Figure 13) and the three events GoodTr, BadTr and GoodDec defined as follows:

- GoodTr holds when $\mathcal{J}' \neq \emptyset$ and $\mathcal{J}' \subseteq \mathcal{J}$, which indicates that at least one real traitor has been traced and no non-traitors were framed by the tracing algorithm.

$\mathsf{Game{-}Trace{-}ID}_{\mathcal{A},\mathcal{E}}(1^\lambda)$

$\mathcal{J} \leftarrow \emptyset$

$(n, t, \mathcal{I}^*, \mathsf{st}_1) \leftarrow\$ \mathcal{A}_1(1^\lambda)$

if $\mathcal{I}^* = \emptyset$ **then** $\mathcal{I}^* \leftarrow\$ \mathsf{IDSpace}$

$(\mathsf{pk}, \{\mathsf{sk}_i\}_{[n]}, \mathsf{tk}) \leftarrow\$ \mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon(\lambda)})$

$(C, \mathsf{st}_2) \leftarrow\$ \mathcal{A}_2(\mathsf{st}_1, \mathsf{pk})$

$D \leftarrow\$ \mathcal{A}_3(\mathsf{st}_2, \{\mathsf{sk}_i\}_{i\in C})$

$\mathcal{J}' \leftarrow\$ \mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, \mathcal{I}^*, 1^{1/\varepsilon(\lambda)})$

return $(\mathsf{pk}, \mathcal{I}^*, D, \mathcal{J}, \mathcal{J}')$

Figure 13: Traceability game for TTIB−KEM**.**

- BadTr holds when at least one non-traitor was framed by the tracing algorithm ($\mathcal{J}' \neq \emptyset$ and $\mathcal{J}' \not\subseteq \mathcal{J}$).
- GoodDec indicates the decoder is good for pk and $\mathcal{I}^*$. More formally, when $P(D) \geq 1/2 + \varepsilon(\lambda)$ where

$$P(D) \coloneqq \Pr\left[D(c, k_b) = b : \begin{array}{c} \mathsf{ID} \leftarrow\$ \mathcal{I}^*, (c, k_0) \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{ID}) \\ k_1 \leftarrow\$ \mathcal{K}_\lambda, b \leftarrow\$ \{0,1\} \end{array}\right].$$

Definition D.3 (Traceability of TTIB−KEM*).* A TTIB−KEM scheme is traceable if there exist a negligible function ν such that TTIB−KEM is $(1/p, \nu)$-traceable for every positive polynomial $p(\lambda)$.

Differences from the definition of [1]. Abdalla et al. [1] define traitor tracing in the identity-based setting. We note that their definition considers a different scenario than ours. In particular, in [1] they consider group-based applications of IBE, e.g., email addresses groups, in which all users of the group can decrypt using a decryption key(s) that associated for the group. Note that the decryption keys for every group are constructed and distributed by a trusted authority who holds the master key. In their model, they want to defend against collisions in which users of the group, using their decryption keys, generate a pirate decoder that is able to decrypt ciphertexts of this specific group. In the tracing definition, the adversary chooses a group ID and the tracing algorithm returns a set of users within group ID that constructed the decoder. In our setting, on the other hand, we consider a scenario where the master key is shared among shareholders, who may collude in order to construct a pirate decoder that is potentially able to decrypt ciphertexts for any identity. Specifically, our definition enables us to trace decoders that are able to decrypt ciphertexts for an identity ID even if the identity key was not derived previously (this is modeled as the set $\mathcal{I}^*$). Observe that this is not captured by the definition in [1] since future identity keys can be derived only using the master key, which is held only by the trusted authority. We emphasize that the possibility to trace decryption for future identity keys is well motivated due to the use case of IBE in threshold mempool encryption, e.g., in [32]. In this application, the identities correspond to epochs and the ability of the adversary to derive future identity keys gains it an advantage in decrypting ciphertexts for future epochs, which allows him to frontrun transactions. As we are motivated mainly by the mempool-privacy application, our traceability definition aims to identify those shareholders, that collude to construct such a decoder.

Remark On identity key derivation oracle. An observant reader may have noted that our traceability game in Figure 13 does not offer an identity key derivation oracle. However, we note that in this game, we consider a stronger adversary who receives enough key shares to derive an identity key for any identity. Indeed, this holds since the adversary receives secret shares for all corrupted parties in C, where $|C| \geq t$. On the other hand, one would consider an extension of our traceability game in which the adversary is offered a *partial identity key derivation oracle*, such that the adversary can choose to receive the partial identity key share from any honest party. This extended traceability notion gives the ability to prove CCA-secure and traceable TT−KEM and TTIB−KEM in the presence of a traceability adversary who has access to partial decryptions. This is interesting, in particular, in encrypted mempool, as it reflects the real setting where the partial decryption shares of the committee are exposed to everyone. That said, we would like to stress that our traceability game is a reasonable model, where we assume that the adversary that attempts to break traceability has access only to the final decryption result (and not the partial decryptions). This reflects a setting where the partial decryption shares are sent to a trusted combiner who aggregates all of them (then the trusted combiner might send a ZK proof for correct aggregation).

We note that currently, it is unclear to us how to simulate the partial identity key oracle in the traceability proof and is left as an interesting open problem.

Remark on selective identity. We highlight the fact that the adversary commits *in the beginning* to $\mathcal{I}^*$ because it somewhat reflects the selective-identity notion that exists for security of IBE schemes in the context of IND−CPA and IND−CCA [11, 22]. For security, it is well established that the normal notions imply the selective-identity notions, but not the other way around [36]. We note that we would be able to achieve traceability without selective-identity using a stronger one-sided security notion for BTIB−KEM where the adversary can decide on the challenge identity *after* learning the public key and secret key shares. While our first construction can achieve this stronger notion, our second construction can not, hence we restrict ourselves to this version of traceability.

D.2 BTIB−KEM to TTIB−KEM Transformation

Next, we present a black-box construction Π for TTIB−KEM from fingerprinting codes and BTIB−KEM. It is very similar to the transformation of [15], but in the identity-based setting. We give the full construction in Figure 14. The construction is black-box with respect to a BTIB−KEM scheme $\Pi^{\mathsf{BTIB-KEM}}$ and a fingerprinting code $\mathcal{F} = (\mathsf{FCGen}, \mathsf{FCTrace})$ as building blocks.

As a direct corollary from the traceability proof of TT−KEM as presented in [15] (which is discussed in Appendix E.3), we achieve the following theorem.

Theorem D.4. *The* TTIB−KEM *construction* Π *is traceable (Definition D.2), if the underlying* BTIB−KEM *protocol* $\Pi^{\mathsf{BTIB-KEM}}$ *is one-sided secure (Definition 4.4) and the fingerprinting code* $\mathcal{F}$ *is fully collusion-resistant.*

$\mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon})$

$\delta \leftarrow \dfrac{1/2\varepsilon}{1/2 - 2/\sqrt{\lambda}}$

$(\Gamma, \ell, \mathsf{tk}) \leftarrow\!\$\ \mathcal{F}.\mathsf{FCGen}(1^n, 2^{-\lambda}, \delta)$

$\left(\mathsf{pk}, \{(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})\}_{i\in[n], j\in[\ell]}\right)$

$\quad \leftarrow\!\$\ \Pi^{\mathsf{BTIB-KEM}}.\mathsf{KGen}(1^\lambda, n, t, \ell)$

for $i \in [n]$ **do**

$\quad w^{(i)} \leftarrow \Gamma_i$

$\quad \mathsf{sk}_i \leftarrow \{\mathsf{sk}_{i,w_j^{(i)}}^{(j)}\}_{j\in[\ell]}$

return $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{tk})$

$\mathsf{Enc}(\mathsf{pk}, \mathsf{ID})$

$j \leftarrow\!\$\ [\ell]$

$(k, (c_0, c_1)) \leftarrow\!\$$

$\quad \Pi^{\mathsf{BTIB-KEM}}.\mathsf{Enc}(\mathsf{pk}, j, \mathsf{ID})$

return $(k, c \coloneqq (c_0, c_1, j))$

$\mathsf{DIdk}(\mathsf{sk}_i, \mathsf{ID})$

$\mathsf{idk}_i \leftarrow\!\$\ \Pi^{\mathsf{BTIB-KEM}}.\mathsf{DIdk}(\mathsf{sk}_i, \mathsf{ID})$

return idk_i

$\mathsf{CombIdk}(S, \{\mathsf{idk}_i\}_{i\in S})$

$\mathsf{idk} \leftarrow \Pi^{\mathsf{BTIB-KEM}}.\mathsf{CombIdk}(S, \{\mathsf{idk}_i\}_{i\in S})$

return idk

$\mathsf{Dec}(\mathsf{idk}, c)$

Parse c as (c_0, c_1, j)

$k \leftarrow \Pi^{\mathsf{BTIB-KEM}}.\mathsf{Dec}(\mathsf{idk}, (c_0, c_1), j)$

return k

$\mathsf{Tr}^{D(\cdot)}(\mathsf{pk}, j, N, \mathcal{I}, (b_k, b_0, b_1))$

$\mathsf{ctr} \leftarrow 0$

for $r = 1, \ldots, N$ **do**

$\quad \mathsf{ID} \leftarrow\!\$\ \mathcal{I}$

$\quad (k^{(0)}, c^{(0)} \coloneqq (c_0^{(0)}, c_1^{(0)})) \leftarrow\!\$\ \Pi^{\mathsf{BTIB-KEM}}.\mathsf{Enc}(\mathsf{pk}, j, \mathsf{ID})$

$\quad (k^{(1)}, c^{(1)} \coloneqq (c_0^{(1)}, c_1^{(1)})) \leftarrow\!\$\ \Pi^{\mathsf{BTIB-KEM}}.\mathsf{Enc}(\mathsf{pk}, j, \mathsf{ID})$

$\quad c^* \leftarrow (c_0^{(b_0)}, c_1^{(b_1)})$

$\quad$ **if** $D(c^*, k^{(b_k)}) = 1$ **then** $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

return ctr

$\mathsf{Trace}^{D(\cdot)}(\mathsf{pk}, \mathsf{tk}, \mathcal{I}, 1^{1/\varepsilon})$

$N \leftarrow \lambda^2$

$B \leftarrow \lambda^{3/2}$

for $j \in [\ell]$ **do**

$\quad p_{001} \leftarrow\!\$\ \mathsf{Tr}^{D(\cdot)}(\mathsf{pk}, j, N, \mathcal{I}, (b_k, b_0, b_1) \coloneqq (0,0,1))$

$\quad p_{100} \leftarrow\!\$\ \mathsf{Tr}^{D(\cdot)}(\mathsf{pk}, j, N, \mathcal{I}, (b_k, b_0, b_1) \coloneqq (1,0,0))$

$\quad p_{111} \leftarrow\!\$\ \mathsf{Tr}^{D(\cdot)}(\mathsf{pk}, j, N, \mathcal{I}, (b_k, b_0, b_1) \coloneqq (1,1,1))$

$\quad a_0 \leftarrow |p_{001} - p_{100}|$

$\quad a_1 \leftarrow |p_{001} - p_{111}|$

$\quad$ **if** $a_0 \geq B$ **then** $w_j \leftarrow 0$

$\quad$ **else if** $a_1 \geq B$ **then** $w_j \leftarrow 1$

$\quad$ **else** $w_j \leftarrow$ '?'

$\bar{w} \leftarrow w_1 w_2 \ldots w_\ell$

$\mathcal{J} \leftarrow \mathcal{F}.\mathsf{FCTrace}(\mathsf{tk}, \bar{w})$

return $\mathcal{J}$

Figure 14: Our black-box TTIB−KEM **construction given a** BTIB−KEM **protocol** $\Pi^{\mathsf{BTIB-KEM}}$ **and a fingerprinting code** $\mathcal{F}$**.**

E Security Proofs of the TT−KEM Construction

E.1 CCA-security of BT−KEM

Proof of Lemma 4.5. Let Game_1 be the IND−CCA game as introduced in Figure 2. We introduce Game_2 by making a modification to the partial decryption oracle O^{PDec}. In particular, we add an additional abort condition that causes the oracle to return $\perp$. Given the queried ciphertext $c \coloneqq ((c_0, c_1), j, \mathsf{vk}_e)$ and the challenge ciphertext $c^* = ((c_0^*, c_1^*), j^*, \mathsf{vk}_e^*)$, the oracle O_2^{PDec} aborts, if $\mathsf{vk}_e = \mathsf{vk}_e^*$ but $c \neq c^*$. The modification is laid out in Figure 15.

Oracle $O_1^{\mathsf{PDec}}((c,\sigma), i)$	Oracle $O_2^{\mathsf{PDec}}((c,\sigma), i)$
Parse c as $((c_0, c_1), j, \mathsf{vk}_e)$	Parse c as $((c_0, c_1), j, \mathsf{vk}_e)$
	if $\mathsf{vk}_e = \mathsf{vk}_e^* \wedge c \neq c^*$ **then**
	$\quad$ **return** $\perp$
if $\Sigma.\mathsf{Verify}(\mathsf{vk}_e, c, \sigma) = 0$ **then**	**if** $\Sigma.\mathsf{Verify}(\mathsf{vk}_e, c, \sigma) = 0$ **then**
$\quad$ **return** $\perp$	$\quad$ **return** $\perp$
if $c = c^*$ **then**	**if** $c = c^*$ **then**
$\quad \mathsf{ctr} \leftarrow\!\$\ \mathsf{ctr} + 1$	$\quad \mathsf{ctr} \leftarrow\!\$\ \mathsf{ctr} + 1$
$\quad$ **if** $\mathsf{ctr} \geq t - \lvert C\rvert$ **then**	$\quad$ **if** $\mathsf{ctr} \geq t - \lvert C\rvert$ **then**
$\quad\quad$ **return** $\perp$	$\quad\quad$ **return** $\perp$
$d_{i,0} \leftarrow \mathsf{PDec}(c, \mathsf{sk}_{i,0})$	$d_{i,0} \leftarrow \mathsf{PDec}(c, \mathsf{sk}_{i,0})$
$d_{i,1} \leftarrow \mathsf{PDec}(c, \mathsf{sk}_{i,1})$	$d_{i,1} \leftarrow \mathsf{PDec}(c, \mathsf{sk}_{i,1})$
return $(d_{i,0}, d_{i,1})$	**return** $(d_{i,0}, d_{i,1})$

Figure 15: Modified O^{PDec} **oracle.**

Lemma E.1. *If* Σ *is* EUF−OT−CMA*-secure following Definition C.2, then* Game_1 *and* Game_2 *are computationally indistinguishable.*

Proof of lemma E.1. We prove lemma E.1 by reducing a successful PPT distinguisher to the unforgeability of Σ. Let $\mathcal{A}$ be a PPT adversary such that

$$\left|\Pr[\mathsf{Game}_{1,\mathcal{A}}^{\mathsf{BT-KEM}}(1^\lambda) = 1] - \Pr[\mathsf{Game}_{2,\mathcal{A}}^{\mathsf{BT-KEM}}(1^\lambda) = 1]\right| \geq \varepsilon(\lambda)$$

for a non-negligible ε. We construct a PPT reduction $\mathcal{B}$ that runs $\mathcal{A}$ internally to win the EUF−OT−CMA game (Figure 10) of Σ with the same probability ε.

Initially, $\mathcal{B}$ receives a challenge-verification key vk^* from the EUF−OT−CMA game. Then, it simulates Game−IND−CCA to $\mathcal{A}$, embedding vk^* as the vk_e^* of the challenge ciphertext. It computes σ^* by querying the signing oracle of the EUF−OT−CMA game on c^*. When $\mathcal{B}$ receives a partial decryption query on $((c,\sigma), i)$ that triggers the additional abort condition of O_2^{PDec} (i.e., $\mathsf{vk}_e = \mathsf{vk}_e^*$ and $c \neq c^*$) and also passes the abort condition $\Sigma.\mathsf{Verify}(\mathsf{vk}_e, c, \sigma)$, then $\mathcal{B}$ outputs (c, σ) as its forgery in the EUF−OT−CMA game.

Analysis. Clearly, $\mathcal{A}$ can only distinguish between the two games by triggering the extra abort condition at least once *while* sending a valid (c, σ). In this case, it holds that $\mathsf{vk}_e = \mathsf{vk}^*$, $c \neq c^*$ and $\Sigma.\mathsf{Verify}(\mathsf{vk}^*, c, \sigma) = 1$, which means that $\mathcal{B}$'s output (c, σ) is a valid forgery, as the message c is new (i.e. not equal to the only signing query c^*) and the signature σ is valid on c. Also, note that $\mathcal{B}$ queries the signing oracle O^{Sign} once. Hence $\mathcal{B}$ breaks EUF−OT−CMA of

Σ with $\mathcal{A}$'s distinguishing advantage

$$\Pr[\text{Game–EUF–OT–CMA}^{\Sigma}_{\mathcal{B}}(1^\lambda) = 1] \geq$$
$$\left|\Pr[\text{Game}^{\text{BT–KEM}}_{1,\mathcal{A}}(1^\lambda) = 1] - \Pr[\text{Game}^{\text{BT–KEM}}_{2,\mathcal{A}}(1^\lambda) = 1]\right| \geq \varepsilon(\lambda)$$

which contradicts the assumption that Σ is EUF–OT–CMA secure. □

We conclude our proof with a reduction from Game_2 to the selective ID semantic security of the underlying BTIB–KEM scheme $\mathcal{I}$.

Let $\mathcal{A}$ be a PPT adversary in Game_2 with $\Pr[\text{Game}^{\text{BT–KEM}}_{2,\mathcal{A}}(1^\lambda) = 1] \geq 1/2 + \varepsilon(\lambda)$ for a non-negligible ε. We construct a PPT reduction $\mathcal{B}$ that wins Game–IND–SID–CPA of $\mathcal{I}$ with the same probability. $\mathcal{B}$ first runs $\mathcal{A}_1$ and receives the parameter ℓ as a result. Then, $\mathcal{B}$ generates a signing keypair $(\mathsf{sk}^*_e, \mathsf{vk}^*_e) \leftarrow\!\$\ \Sigma.\text{KGen}(1^\lambda)$ and returns vk^*_e as the selected challenge-identity $\mathcal{I}^*$ as well as ℓ to Game–IND–SID–CPA. In the next step, $\mathcal{B}$ receives the public key pk from Game–IND–SID–CPA, forwards it to $\mathcal{A}_2$ and sends the corrupted set C back to Game–IND–SID–CPA. Next, $\mathcal{B}$ receives a challenge ciphertext $c^*_{IBE} = (c^*_0, c^*_1)$, the challenge key k^* and the challenge index j^* as well as the corrupted secret key shares $\{\mathsf{sk}_i\}_{i\in C}$. $\mathcal{B}$ converts this into a BT–KEM challenge ciphertext $c^* = ((c^*_0, c^*_1), j^*, \mathsf{vk}^*_e)$ and computes $\sigma^* \leftarrow\!\$\ \text{GenTag}(c^*, \mathsf{sk}^*_e)$. It forwards $(c^*, \sigma^*), k^*$ and $\{\mathsf{sk}_i\}_{i\in C}$ to $\mathcal{A}_3$ and returns the same bit as $\mathcal{A}_3$.

Simulating O^{PDec}_2. $\mathcal{B}$ simulates O^{PDec}_2 by querying the identity-key derivation oracle O^{DIdk} of Game–IND–SID–CPA on the identity vk_e for the PDec step of O^{PDec}_2. Observe that the internal counter $\text{ctr}_{O^{\text{PDec}}_2}$ of O^{PDec}_2 is always greater or equal to the internal counter $\text{ctr}_{O^{\text{DIdk}}}$ of the O^{DIdk} oracle because O^{PDec}_2 aborts whenever $\mathsf{vk}_e = \mathsf{vk}^*_e = \mathcal{I}^*$, before even querying O^{DIdk} on $\mathcal{I}^*$. Hence, it can never happen that the O^{DIdk} oracle hits its abort condition of $\text{ctr}_{O^{\text{DIdk}}} \geq t - |C|$ and the simulation is always successful.

Analysis. Let $\hat{b}$ be the internal bit of Game–IND–SID–CPA and b the internal bit of Game_2. If $\hat{b} = 1$, then $\mathcal{B}$ clearly simulates Game_2 for $b = 1$ and vice versa. Consequently, $\mathcal{B}$'s success probability in Game–IND–SID–CPA is equal to $\mathcal{A}$'s success probability in Game_2 and we get

$$\Pr[\text{Game–IND–SID–CPA}^{\mathcal{I}}_{\mathcal{B}}(1^\lambda) = 1] =$$
$$\Pr[\text{Game}^{\text{BT–KEM}}_{2,\mathcal{A}}(1^\lambda) = 1] \geq 1/2 + \varepsilon(\lambda)$$

which is a contradiction to the IND–SID–CPA security of $\mathcal{I}$.

Overall, we can bound the advantage of any PPT adversary $\mathcal{A}$ in Game–IND–CCA of BT–KEM as follows.

$$\text{Adv}^{\text{BT–KEM}}_{\mathcal{A},\text{Game–IND–CCA}}$$
$$\leq \text{Adv}^{\Sigma}_{\mathcal{B},\text{Game–EUF–OT–CMA}} + \text{Adv}^{\mathcal{I}}_{\mathcal{B},\text{Game–IND–SID–CPA}}$$

Hence the advantage is negligible by our assumption that Σ is EUF–OT–CMA secure and $\mathcal{I}$ is IND–SID–CPA secure □

E.2 One-Sided Security of BT–KEM

Proof of Lemma 4.6. Let $\mathcal{A}$ be a successful adversary against Game–OSS of BT–KEM (Figure 3) such that $\Pr[\text{Game–OSS}^{\text{BT–KEM}}_{\mathcal{A}}(1^\lambda) = 1] \geq 1/2 + \varepsilon(\lambda)$ for a non-negligible ε. We construct a PPT reduction $\mathcal{B}$ to Game–OSS–ID (Figure 6) of the underlying BTIB–KEM scheme $\mathcal{I}$. First, the reduction runs $\mathcal{A}_1$ and receives ℓ, u and d. Then it obtains a signing key-pair $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\!\$\ \Sigma.\text{KGen}(1^\lambda)$, sets $\text{ID} \leftarrow \mathsf{vk}_e$ and passes l, u, d and ID to Game–OSS–ID. As a result, it receives the secret-key shares S, the public key pk, the challenge ciphertext $c^* = (c^*_0, c^*_1)$, and the challenge key k. $\mathcal{B}$ transforms this into a challenge ciphertext of Game–OSS, by setting $c^*_{\text{OSS}} \coloneqq ((c^*_0, c^*_1), j, \mathsf{vk}_e)$ and computing $\sigma^* \leftarrow\!\$\ \text{GenTag}(\mathsf{sk}_e, c^*_{\text{OSS}})$. It forwards pk, $(c^*_{\text{OSS}}, \sigma^*)$, k, and S to $\mathcal{A}_2$ and returns the same bit as $\mathcal{A}_2$.

$\mathcal{B}$'s success probability in Game–OSS–ID is equal to $\mathcal{A}$'s success probability in Game–OSS, as $\mathcal{B}$ perfectly simulates Game–OSS for $b = 1$, whenever the internal bit of Game–OSS–ID is also 1 and vice versa. Hence,

$$\Pr[\text{Game–OSS–ID}^{\mathcal{I}}_{\mathcal{B}}(1^\lambda) = 1] =$$
$$\Pr[\text{Game–OSS}^{\text{BT–KEM}}_{\mathcal{A}}(1^\lambda) = 1] \geq 1/2 + \varepsilon(\lambda)$$

which is a contradiction to the one-sided security of $\mathcal{I}$. □

E.3 Traceability Proof of TT–KEM

Here, we present in detail, how the traceability proof of [15] can be adapted to work for our modified TT–KEM transformation from our definition of CCA-secure BT–KEM. The crucial differences here are (a) the modifications to the one-sided security game Game–OSS of BT–KEM (see Figure 3) and (b) the modified Tr subroutine of the TT–KEM construction (see Figure 4).

Recall briefly, that the tracing works by running the $\text{Tr}^{D(\cdot)}$ subroutine for every index $j \in \ell$ with either valid ciphertexts or ciphertexts that are malformed in some way. Afterwards, for every j the results of the different Tr runs are compared to determine, whether the decoder contains only left keys for j, only right keys for j, or mixed keys. The results are compiled into a noisy codeword $\bar{w}$, which is essentially just a string of length ℓ, where 0 encodes "only left keys at positions j", 1 encodes only right keys at position j, and '?' encodes that the tracing algorithm can not determine whether the decoder has only left, only right, or mixed keys at that position. Henceforth, the '?' positions are called *noisy*. In the end, the codeword $\bar{w}$ is fed into the tracing algorithm of the fingerprinting code along with the tracing key tk to determine the set of traitors.

The proof in [15] can be separated into four phases. We next discuss these phases and explain why they are not affected by our modifications.

In the first phase they argue that the codeword $\bar{w}$ produced by the tracing algorithm for any *good* decoder can at most have a fraction of δ noisy positions with overwhelming probability[11]. This is a necessary precondition for the tracing algorithm of the fingerprinting code to work. This lemma follows from purely probabilistic arguments that are derived from the definition of good decoders, hence they are entirely unaffected by our modifications.

In the second phase, they introduce an idealized tracing experiment, where they instead use an idealized tracing algorithm Trace′ that depends on the *actual* set of traitors $\mathcal{J}$, i.e. they call Tr^D with different (b_k, b_0, b_1) dependent on whether all real traitors have left

[11] The parameter $\delta \in [0, 1]$ is the noisiness limit of the fingerprinting code and it is fixed depending on the desired tracing parameter ε in the TT–KEM construction.

keys at position j. They argue that the idealized tracing algorithm outputs a codeword $\bar{w}'$ that is feasible for the set of traitors $\mathcal{J}$ with overwhelming probability[12][13]. The arguments made to prove this lemma rely on the fact that $\mathsf{Tr}^D((b_k, b_0, b_1) = (0,1,1))$ and $\mathsf{Tr}^D((b_k, b_0, b_1) = (1,0,0))$ are identically distributed. Observe that this is equally the case for our modified Tr subroutine, hence the proof for this lemma is unaffected.

Now that we have the two facts that (1) $\bar{w}$ and $\bar{w}'$ are δ-noisy and (2) that $\bar{w}'$ is feasible for $\mathcal{J}$, it remains to show that the results of the real and the idealized tracing $\bar{w}$ and $\bar{w}'$ are computational indistinguishably distributed. This is carried out in phases three and four through a reduction to one-sided security. Essentially, they argue that any adversary that can meaningfully distinguish between the two experiments, can be transformed to break the one-sided security game of BT–KEM.

First, the one-sided security game is extended to a more general version, which outputs N challenges for a common bit b instead of just one challenge. This is carried out through a standard hybrid argument, which can be applied to our modified Game–OSS in exactly the same manner. In particular, we get the following bound on the advantage in the generalized game

$$\mathbf{Adv}_{\mathcal{A}}^{\text{Game–OSS-gen}} \leq N \cdot \mathbf{Adv}_{\mathcal{A}}^{\text{Game–OSS}}.$$

Finally, they relate the probabilities that $\bar{w}$ and $\bar{w}'$ are feasible for $\mathcal{J}$ by bounding

$$\left|\Pr[\bar{w} \text{ is not feasible for } \mathcal{J}] - \Pr[\bar{w}' \text{ is not feasible for } \mathcal{J}]\right|$$

by the one-sided security of BT–KEM. They reduce any distinguishing adversary $\mathcal{A}$ to the generalized one-sided security game. The reduction $\mathcal{B}$ works by picking a random j^* and d and embedding the N challenge ciphertexts received by Game–OSS-gen as the ciphertexts that would be generated by the Tr subroutine that is called with different (b_k, b_0, b_1) in the real versus the idealized experiment. If the internal bit b of Game–OSS-gen is 0, then the reduction simulates the real tracing experiment. If, however, the internal bit b is 1, then the reduction's simulation amounts to the idealized game. The reduction answers with the bit $\bar{w}_{j^*}$ or outputs a random bit, should $\mathcal{A}$ corrupt the wrong side for j^*. In the end, the reduction achieves the following bound:

$$\begin{aligned}&\left|\Pr[\bar{w} \text{ is not feasible for } \mathcal{J}] - \Pr[\bar{w}' \text{ is not feasible for } \mathcal{J}]\right| \\ &\leq 2\ell \mathbf{Adv}_{\mathcal{B}}^{\text{Game–OSS-gen}} \\ &\leq 2\ell N \mathbf{Adv}_{\mathcal{B}}^{\text{Game–OSS}}.\end{aligned}$$

To see, why the same reduction works for our modifications, we make a close comparison between the (modified) Tr subroutine and the (modified) Game–OSS in Figure 16. One can clearly see that the modifications to Tr and Game–OSS mirror each other. Specifically, the only difference over [15] is the additional tag τ^* that is given to the decoder (resp. the adversary) and it is generated in exactly the same manner in both Tr and Game–OSS. Hence, when $\mathcal{B}$ uses the ciphertext-tag pairs as generated by our modified one-sided security game to simulate the corresponding ciphertexts within the modified Tr subroutine, $\mathcal{B}$ equally simulates either the real tracing experiment or the idealized tracing experiment depending on the interal bit b of Game–OSS.

[12]Essentially, $\bar{w}$ is feasible for $\mathcal{J}$, if $\bar{w}$ does not contain a 1 at position j, where the parties in $\mathcal{J}$ have all left keys and vice versa.

[13]The feasibility of $\bar{w}$ for $\mathcal{J}$ is important, because the tracing algorithm of the fingerprinting code that is run in the end is guaranteed to output a non-empty subset of $\mathcal{J}$, if $\bar{w}$ is feasible for $\mathcal{J}$ (and $\bar{w}$ is δ-noisy).

F Security of BTIB–KEM–1

F.1 Semantic Security

We prove semantic security in two steps. First, we introduce a new augmented bilinear decisional Diffie Hellman (ABDDH) assumption that is easier to work with, but is reducible to standard BDDH (Definition C.4). We refer to Appendix G for the full reduction.

Definition F.1 (Augmented Bilinear Decisional Diffie-Hellman). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the augmented bilinear decisional Diffie-Hellman (ABDDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries it holds that

$$\left|\Pr\left[\mathcal{A}(E, \vec{x}, e(g_1, g_2)^{w\alpha yzr}) = 1\right] - \Pr\left[\mathcal{A}(E, \vec{x}, e(g_1, g_2)^{v}) = 1\right]\right|$$

is negligible in λ where $E \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$, $w, \alpha, y, z, r, v \leftarrow\!\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \left(g_1, g_1^y, g_1^z, g_1^{yz}, g_1^{yr}, g_1^{zr}, g_1^{\alpha yz}, g_2, g_2^y, g_2^z, g_2^w, g_2^{y\alpha}, g_2^{z\alpha}\right).$$

We conclude the proof with a reduction from IND–SID–CPA-security of BTIB–KEM–1 to the ABDDH assumption.

Proof of Lemma 5.1. Let $\ell_{max} > 0$ be an upper bound for ℓ, which is an arbitrary but fixed polynomial in the security parameter such that $1 \leq \ell \leq \ell_{max}$. We prove IND–SID–CPA security of our first BTIB–KEM construction by reduction to the ABDDH assumption (Definition F.1). Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be a PPT adversary against the Game–IND–SID–CPA experiment (Figure 5) such that

$$\Pr[\text{Game–IND–SID–CPA}_{\mathcal{A}}^{\text{BTIB–KEM}}(1^\lambda) = 1] \geq 1/2 + \varepsilon(\lambda)$$

for a non-negligible ε. We construct a PPT reduction $\mathcal{B}$ that uses $\mathcal{A}$ internally to break the ABDDH assumption.

First, $\mathcal{B}$ receives the security parameter 1^λ as well as an ABDDH instance

$$(g_1, g_1^y, g_1^z, g_1^{yz}, g_1^{yr}, g_1^{zr}, g_1^{\alpha yz}), (g_2, g_2^y, g_2^z, g_2^w, g_2^{y\alpha}, g_2^{z\alpha}), T$$

where the challenge T is either $e(g_1, g_2)^{w\alpha yzr}$ or $e(g_1, g_2)^v$ for a random $v \in \mathbb{Z}_p$.

$\mathcal{B}$ samples a random $j^* \in [\ell_{max}]$ and runs $\mathcal{A}_1(1^\lambda)$ to receive the selected challenge identity $\mathcal{I}^*$ and the number of positions ℓ, while simulating the random oracle.

Simulating the random oracle $\mathcal{H}(j, \mathsf{ID})$. For each query $\mathcal{H}(j, \mathsf{ID})$, $\mathcal{B}$ simulates the random oracle $\mathcal{H}$ as follows: If $(j, \mathsf{ID}) = (j^*, \mathcal{I}^*)$, then output the group element g_2^w from the ABDDH instance. Otherwise, sample a random $w_{j,\mathsf{ID}} \leftarrow\!\!\$\ \mathbb{Z}_p$ and output $g_2^{w_{j,\mathsf{ID}}}$. $\mathcal{B}$ answers queries *consistently*, i.e. it saves the values and repeats answers for identical queries.

If $j^* \notin \ell$, then $\mathcal{B}$ stops the simulation and outputs a random bit. One can bound the loss incurred here by $1/\ell_{max}$. Otherwise, $\mathcal{B}$ simulates KGen as follows:

- For $j \neq j^*$, run the normal key generation procedure.
- For j^*, embed the ABDDH instance into the public key. In particular, set $X_{j^*} \leftarrow g_1^{\alpha yz}$ and set $Y_{j^*} \leftarrow g_1^y$ as well as $Z_{j^*} \leftarrow g_1^z$, which are part of the ABDDH instance.

$\mathsf{Tr}^{D(\cdot)}(\mathsf{pk}, j, N, b_k, b_0, b_1)$

$\mathsf{ctr} \leftarrow 0$
for $r = 1, \ldots, N$ **do**
  $(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\!\$\ \mathsf{TagKeys}(1^\lambda)$
  $(k^{(0)}, c^{(0)} \coloneqq (c_0^{(0)}, c_1^{(0)})) \leftarrow\!\$\ \mathsf{BT{-}KEM.Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$
  $(k^{(1)}, c^{(1)} \coloneqq (c_0^{(1)}, c_1^{(1)})) \leftarrow\!\$\ \mathsf{BT{-}KEM.Enc}(\mathsf{pk}, j, \mathsf{vk}_e)$
  $c^* \leftarrow (c_0^{(b_0)}, c_1^{(b_1)})$
  $\pi^* \leftarrow\!\$\ \mathsf{GenTag}(c^*, \mathsf{sk}_e)$
  if $D((c^*, \pi^*), k^{(b_k)}) = 1$ **then** $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
return ctr

$\mathsf{Game{-}OSS}_{\mathcal{A}}(1^\lambda)$

$(\ell, u, d, \mathsf{st}_1) \leftarrow\!\$\ \mathcal{A}_1(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda, n, t, \ell)$
$(\mathsf{sk}_e, \mathsf{vk}_e) \leftarrow\!\$\ \mathsf{TagKeys}(1^\lambda)$
$(k^{(0)}, c^{(0)} \coloneqq (c_0^{(0)}, c_1^{(0)})) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, u, \mathsf{vk}_e)$
$(k^{(1)}, c^{(1)} \coloneqq (c_0^{(1)}, c_1^{(1)})) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, u, \mathsf{vk}_e)$
$b \leftarrow\!\$\ \{0, 1\}$
if $b = 0$ **then** $c^* \leftarrow (c_0^{(0)}, c_1^{(1)})$
if $b = 1$ **then** $c^* \leftarrow (c_0^{(d)}, c_1^{(d)})$
$\pi^* \leftarrow\!\$\ \mathsf{GenTag}(c^*, \mathsf{sk}_e)$
$\mathsf{shares} \leftarrow \{(\mathsf{sk}_{i,0}^{(j)}, \mathsf{sk}_{i,1}^{(j)})\}_{i\in[n], j\in[\ell]\setminus\{u\}}, \{\mathsf{sk}_{i,d}^{(u)}\}_{i\in[n]}$
$b' \leftarrow\!\$\ \mathcal{A}_2(\mathsf{st}_1, \mathsf{pk}, (c^*, \pi^*), k^{(0)}, \mathsf{shares})$
return $b \stackrel{?}{=} b'$

Figure 16: Comparison between the Tr **subroutine and the one-sided security game of** BT–KEM**. Modifications over [15] are highlighted in gray.**

$\mathcal{B}$ runs $\mathcal{A}_2$ with the resulting public key and receives a set C of corrupted parties. Then, $\mathcal{B}$ sets the challenge encapsulation key k to the ABDDH challenge $k \leftarrow T$ and the challenge ciphertext $c \leftarrow (g_1^{yr}, g_1^{zr})$, which $\mathcal{B}$ also takes from the ABDDH instance. Finally, $\mathcal{B}$ assembles the secret key of the corrupted parties $\{\mathsf{sk}_i\}_{i\in C}$ as follows. For $j \neq j^*$, $\mathcal{B}$ uses the normal secret keys from KGen. For j^*, $\mathcal{B}$ just samples a random left and right key from $\mathbb{Z}_p$ on behalf of every corrupted party. W.l.o.g. we assume that $|C| = t - 1$. Let u_i be the left secret key of corrupted party i and v_i be the right secret key of corrupted party i. Note that u_i corresponds to the field element $z \cdot s_i^{(j^*)}$ and v_i corresponds to $y \cdot s_i^{(j^*)}$. It is possible to simulate these with random values, as $\mathcal{A}$ only receives up to $t - 1$ shares.

$\mathcal{B}$ runs $\mathcal{A}_3^{O^{\mathsf{DIdk}}}((c, k), \{\mathsf{sk}_i\}_{i\in C})$, receiving a bit b' as a result. In the end, $\mathcal{B}$ returns b' as its guess in the ABDDH instance (if $b' = 0$, it guesses that the challenge T is equal to $e(g_1, g_2)^v$ (i.e., unifromly random in $\mathbb{G}_T$) and if $b' = 1$, it guesses that the challenge is equal to $e(g_1, g_2)^{w\alpha yzr}$). On top of that, $\mathcal{B}$ needs to simulate oracle queries to O^{DIdk} to $\mathcal{A}$.

Simulating the identity key derivation oracle $O^{\mathsf{DIdk}}(\mathsf{ID}, x)$. If the target party x is corrupted ($x \in C$), we can trivially simulate identity key derivation using the simulated key shares u_x and v_x. Otherwise, we do the following: For all $j \neq j^*$, $\mathcal{B}$ can simulate O^{DIdk} according to protocol, as it knows all the secret keys. For j^*, we distinguish between 2 cases. If $\mathsf{ID} = \mathcal{I}^*$, then we abort[14]. If $\mathsf{ID} \neq \mathcal{I}^*$ we do the following:

- Define the polynomial f with $f(0) = w_{j^*,\mathsf{ID}} \cdot z\alpha$ and $f(i) = u_i$ for $i \in C$.
- Compute $g_2^{f(0)} \leftarrow (g_2^{z\alpha})^{w_{j^*,\mathsf{ID}}}$ from the ABDDH instance[15] and $g_2^{f(i)} \leftarrow g_2^{u_i}$ for $i \in C$. Use these t values to interpolate f in $\mathbb{G}_2$ and evaluate $g_2^{f(x)} \leftarrow \Pi_{i\in C\cup\{0\}}(g_2^{f(i)})^{l_{i,C\cup\{0\}}(x)}$ where $l_{i,C\cup\{0\}}(x)$ is the Lagrange coefficient of index i for interpolation at position x among the set $C \cup \{0\}$.
- Set $\mathsf{idk}_{i,0}^{(j^*)} \leftarrow g_2^{f(x)}$.
- Repeat the same for the right key for the function g with $g(0) = w_{j^*,\mathsf{ID}} \cdot y\alpha$ and $g(i) = v_i$ for $i \in C$.

Analysis. Observe that if $\mathcal{B}$ is running in an ABDDH instance where v is random, then $\mathcal{B}$ simulates Game–IND–SID–CPA with $b = 0$ to $\mathcal{A}$. If, however, the ABDDH challenge equals $e(g_1, g_2)^{w\alpha yzr}$, then $\mathcal{B}$ simulates Game–IND–SID–CPA with $b = 1$. Hence, $\mathcal{B}$ has the non-negligible advantage $\varepsilon(\lambda)/\ell_{max}$ to break ABDDH. □

F.2 One-Sided Security

Just as in the semantic security proof, we introduce the following augmented XDH assumption (AXDH) that is easier to work with in the security proof. A full reduction from AXDH to standard XDH (Definition C.5) is presented in Appendix G.

Definition F.2 (Augmented XDH Assumption). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the augmented external Diffie-Hellman (AXDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries it holds that

$$\left|\Pr\left[\mathcal{A}(E, \vec{x}, g_1^{zr}) = 1\right] - \Pr\left[\mathcal{A}(E, \vec{x}, g_1^{zr'}) = 1\right]\right|$$

is negligible in λ where $E \leftarrow\!\$\ \mathcal{G}(1^\lambda)$, and $y, z, r, r' \leftarrow\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \left(g_1, g_1^y, g_1^z, g_1^{yr}, g_1^{yr'}, g_2\right).$$

[14] This is what would happen in the game, as we assume that $|C| = t - 1$. If $|C| < t - 1$, the simulation could return two random group elements *consistently* instead.

[15] This is possible, because for $\mathsf{ID} \neq \mathcal{I}^*$ the field element $w_{j^*,\mathsf{ID}}$ is sampled by $\mathcal{B}$ during simulation of the random oracle and thus known to $\mathcal{B}$. In case $\mathcal{A}$ has never queried for $\mathcal{H}(j^*, \mathsf{ID})$, $\mathcal{B}$ just samples a value for $w_{j^*,\mathsf{ID}}$ and continues to answer consistently for the random oracle.

Our result of one-sided security is therefore captured by the following Lemma.

Lemma F.3. *For all PPT adversaries $\mathcal{A}$, there exists a PPT reduction $\mathcal{B}$ such that*

$$\mathsf{Game{-}OSS{-}ID}_{\mathcal{A}}^{\mathsf{BTIB-KEM-1}}(1^\lambda) \leq \mathbf{Adv}_{\mathcal{B},\mathcal{G}}^{\mathsf{AXDH}}$$

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

Proof of Lemma F.3. We make a reduction from Game−OSS−ID of BTIB−KEM−1 (Figure 6) to AXDH (Definition F.2). Let $\mathcal{A}$ be a PPT adversary such that $\Pr[\mathsf{Game{-}OSS{-}ID}_{\mathcal{A}}^{\mathsf{BTIB-KEM-1}}(1^\lambda) = 1] \geq 1/2 + \varepsilon(\lambda)$. We construct a reduction $\mathcal{B}$, which is solving the AXDH problem. $\mathcal{B}$ receives $g_1, g_1^y, g_1^z, g_1^{yr}, g_1^{yr'} \in \mathcal{G}_1$ and $g_2 \in \mathcal{G}_2$ as input along with a challenge T which is either g_1^{zr} or $g_1^{zr'}$.

$\mathcal{B}$ runs $\mathcal{A}$ to get $(\ell, u, d, \mathsf{ID}^*)$, while simulating the random oracle as follows throughout the reduction. For now, we assume that $d = 0$. At the end of the proof, we explain what changes for $d = 1$. In the $d = 0$ case, the adversary $\mathcal{A}$ gets a matching key k and left ciphertext c_0 and needs to distinguish whether the right ciphertext c_1 fits with k and c_0 (case $b = 1$) or belongs to another encryption (case $b = 0$). In the reduction, we will leverage $\mathcal{A}$'s advantage by embedding the challenge T as the right ciphertext.

Simulating the random oracle $\mathcal{H}$. For each query $\mathcal{H}(j, \mathsf{ID})$, $\mathcal{B}$ simulates the random oracle $\mathcal{H}$ by sampling a random $w_{j,\mathsf{ID}} \leftarrow\!\!\$\ \mathbb{Z}_p$ and outputting $g_2^{w_{j,\mathsf{ID}}}$. $\mathcal{B}$ answers queries *consistently*, i.e. it saves the values and repeats answers for identical queries. We assume, $\mathcal{A}$ makes a hash query on the target identity ID^* for level u; if not, the reduction makes the query $\mathcal{H}(u, \mathsf{ID}^*)$ itself.

Simulating key generation. Our simulation differs between the target position u and all other positions as follows:

- For all $j \neq u$, $j \in [\ell]$, we sample α_j, y_j, z_j, compute the public keys X_j, Y_j, Z_j and all secret key shares for both sides honestly.
- For $j = u$, we need to simulate only left key shares. For this, we sample $\beta \leftarrow\!\!\$\ \mathbb{Z}_p$ and embed $X_u \leftarrow \left(g_1^y\right)^\beta$, as well as $Y_u \leftarrow g_1^y$ and $Z_u \leftarrow g_1^z$ from the AXDH instance. Following the notation of the scheme, we view $X_u = g_1^{\alpha yz}$ where $\alpha = \beta/z$ or $\beta = \alpha z$. Next, we create a Shamir secret sharing $\beta_1, \ldots, \beta_n$ of β. We set $sk_{i,0}^{(u)} = \beta_i$

Simulating the key and ciphertext. Let $w = w_{u,ID^*}$. Recall that $\mathcal{B}$ receives g_1^{yr} as part of the challenge. $\mathcal{B}$ computes $k_0 \leftarrow e\left(g_1^{yr}, g_2^w\right)^\beta = e\left(g_1, g_2\right)^{\alpha wyzr}$. We set the left ciphertext as $c_0^{(0)} \leftarrow g_1^{yr}$ and the right ciphertext as $c_1 \leftarrow T$, where T is AXDH challenge. We pass k_0, c_0 and c_1 together with the secret shares and the public key to the adversary. In the end, the reduction $\mathcal{B}$ outputs the same bit as $\mathcal{A}$.

Analysis. When $T = g_1^{zr}$, then $\mathcal{B}$ simulates all the parameters for $b = 1$ as $c_1 = T = g_1^{zr}$ fits with the pair $k_0 = e(X_u, \mathcal{H}(u, \mathsf{ID}^*))^r$ and $c_0^{(0)} = g_1^{yr}$. Similarly when $T = g_1^{zr'}$ then $\mathcal{B}$ simulates all the parameters for $b = 0$, where the right ciphertext $c_1 = g_1^{zr'}$ does not fit with k_0 and $c_0^{(0)}$. Hence, $\mathcal{B}$ has the same advantage against AXDH, as $\mathcal{A}$ against Game−OSS−ID of BTIB−KEM−1.

***Case* d = 1.** For this case, we swap Y_u and Y_z in the public key. We essentially treat g_1^y as g_1^z and vice-versa. We set the challenge as follows:

- Set $c_1 = g_1^{yr}$, $c_0 = T$, where T is the challenge $\mathcal{B}$ received. $T \in \{g_1^{zr}, g_1^{zr'}\}$.
- Set $c^* = (c_0, c_1)$
- Let $w = w_{u,ID^*}$. Recall, $\mathcal{B}$ receives $g_1^{yr'}$ as part of the challenge. $\mathcal{B}$ computes

$$k_0 = \left(g_1^{yr'}, g_2^w\right)^\beta = (g_1, g_2)^{\alpha wyzr'}.$$

The correctness of the decapsulation follows similarly as above. Clearly, k_0 does not fit with c_1. When $T = g_1^{zr}$ then $\mathcal{B}$ simulates all the parameters for $b = 1$. In particular, $c_0 = g_1^{zr}$ matches with $c_1 = g_1^{yr}$. Similarly when $T = g_1^{zr'}$ then $\mathcal{B}$ simulates all the parameters for $b = 0$ as $c_0 = g_1^{zr'}$ matches with $k_0 = e\,(g_1, g_2)^{\alpha wyzr'}$. This concludes the proof. □

G Reductions to Well-Established Assumptions

In this section, we reduce the ABDDH assumption and the AXDH assumption, which were introduced to make a more comprehensible proof of semantic security and one-sided security for BTIB−KEM−1, to the standard BDDH and XDH assumptions

First we recall the two augmented hardness assumptions.

Definition G.1 (Augmented Bilinear Decisional Diffie-Hellman). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the augmented bilinear decisional Diffie-Hellman (ABDDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries it holds that

$$\left|\Pr\left[\mathcal{A}(E, \vec{x}, e(g_1, g_2)^{w\alpha yzr}) = 1\right] - \Pr\left[\mathcal{A}(E, \vec{x}, e(g_1, g_2)^v) = 1\right]\right|$$

is negligible in λ where $E \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$, $w, \alpha, y, z, r, v \leftarrow\!\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \left(g_1, g_1^y, g_1^z, g_1^{yz}, g_1^{yr}, g_1^{zr}, g_1^{\alpha yz}, g_2, g_2^y, g_2^z, g_2^w, g_2^{y\alpha}, g_2^{z\alpha}\right).$$

The second one is the augmented XDH assumption

Definition G.2 (Augmented XDH Assumption). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the augmented external Diffie-Hellman (AXDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries it holds that

$$\left|\Pr\left[\mathcal{A}(E, \vec{x}, g_1^{zr}) = 1\right] - \Pr\left[\mathcal{A}(E, \vec{x}, g_1^{zr'}) = 1\right]\right|$$

is negligible in λ where $E \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$, and $y, z, r, r' \leftarrow\!\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \left(g_1, g_1^y, g_1^z, g_1^{yr}, g_1^{yr'}, g_2\right).$$

G.1 Augmented BDDH

We show a reduction from the standard Bilinear Decisional Diffie-Hellman problem for type-3 pairing [20] as defined in Definition C.4. We redefine it here with different variable names to make the reduction easier to follow.

Definition G.3 (Bilinear Decisional Diffie-Hellman). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the bilinear decisional Diffie-Hellman (BDDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries $\mathcal{B}$, it holds that

$$\left|\Pr\left[\mathcal{B}(E,\vec{x},e(g_1,g_2)^{w\alpha r})=1\right]-\Pr\left[\mathcal{B}(E,\vec{x},e(g_1,g_2)^{v})=1\right]\right|$$

is negligible in λ where $E \leftarrow\!\$\ \mathcal{G}(1^\lambda)$, $w,\alpha,r,v \leftarrow\!\$\ \mathbb{Z}_p$ and

$$\vec{x}=\left(g_1,g_1^{\alpha},g_1^{r},g_2,g_2^{w},g_2^{\alpha}\right).$$

The reduction works in the following way. Suppose we are given an adversary $\mathcal{A}$ against the Augmented BDDH problem. We design an adversary $\mathcal{B}$ against the standard BDDH problem. The input to $\mathcal{B}$ is $(E,\vec{\chi},\hat{T})$ where either $\hat{T}=e(g_1,g_2)^{w\alpha r}$ (real case) or $\hat{T}=e(g_1,g_2)^{v}$ (ideal case). $\mathcal{B}$ samples $y,z \leftarrow\!\$\ \mathbb{Z}_p$ and computes the following elements of $\mathbb{G}_1$ in $\vec{x}$.

$$g_1^{y},g_1^{z},g_1^{yz},\left(g_1^{r}\right)^{y},\left(g_1^{r}\right)^{z},\left(g_1^{\alpha}\right)^{yz}.$$

For the elements in $\mathbb{G}_2$, $\mathcal{B}$ already got g_2^{w} as the input. For the rest of the elements of $\mathbb{G}_2$ in $\vec{x}$, $\mathcal{B}$ computes

$$g_2^{y},g_2^{z},\left(g_2^{\alpha}\right)^{y},\left(g_2^{\alpha}\right)^{z}.$$

Finally, $\mathcal{B}$ computes $T=\hat{T}^{yz}$ and invokes $\mathcal{A}$ on $(E,\vec{x},T)$. If $\mathcal{A}$ returns real, $\mathcal{B}$ outputs real. Otherwise, $\mathcal{B}$ outputs random.

For the analysis, we see that all the elements of $\vec{x}$ is computed following the same distribution as in the problem statement. For real case, T is computed truthfully. Further, as $\hat{v}$ is uniformly distributed and independent from y,z, the distribution of $v=\hat{v}yz$ is uniform. Thus distribution of T is identical to that of the problem statement. Hence $\mathcal{B}$ simulates the problem instance perfectly.
We conclude

$$\mathbf{Adv}^{\mathsf{BDDH}}_{\mathcal{B},\mathcal{G}}=\mathbf{Adv}^{\mathsf{ABDDH}}_{\mathcal{A},\mathcal{G}}$$

G.2 Augmented XDH

Recall the standard XDH assumption. We redefine it with different variable names.

Definition G.4 (XDH Assumption). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the external Diffie-Hellman (XDH) problem is hard for $\mathcal{G}$, if for all PPT adversaries $\hat{\mathcal{B}}$ it holds that

$$\left|\Pr\left[\mathcal{B}(E,\vec{x},g_1^{zr})=1\right]-\Pr\left[\mathcal{B}(E,\vec{x},g_1^{v})=1\right]\right|$$

is negligible in λ where $E \leftarrow\!\$\ \mathcal{G}(1^\lambda)$, and $z,r,v \leftarrow\!\$\ \mathbb{Z}_p$ and

$$\vec{x}=\left(g_1,g_1^{z},g_1^{r},g_2\right).$$

We show a two step reduction from the standard XDH problem. First, we introduce the external 2-Diffie Hellman problem.

Definition G.5 (X2DH Assumption). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the external two Diffie-Hellman (X2DH) problem is hard for $\mathcal{G}$, if for all PPT adversaries $\mathcal{B}$ it holds that

$$\left|\Pr\left[\mathcal{B}(E,\vec{\chi},g_1^{zr})=1\right]-\Pr\left[\mathcal{B}(E,\vec{\chi},g_1^{zr'})=1\right]\right|$$

is negligible in λ where $E \leftarrow\!\$\ \mathcal{G}(1^\lambda)$, and $z,r,r' \leftarrow\!\$\ \mathbb{Z}_p$ and

$$\vec{\chi}=\left(g_1,g_1^{z},g_1^{r},g_1^{r'},g_2\right).$$

Hardness of X2DH can be established by applying the XDH assumption twice (once between g_1^{zr} and g_1^{v} and then between $g_1^{zr'}$ and g_1^{v}). Thus for all PPTM $\mathcal{B}$, it holds that

$$\mathbf{Adv}^{\mathsf{X2DH}}_{\mathcal{B},\mathcal{G}} \le 2\mathbf{Adv}^{\mathsf{XDH}}_{\mathcal{B},\mathcal{G}}$$

For the second step, we show a reduction from the AXDH problem to the X2DH problem. Consider any adversary $\mathcal{A}$ against the AXDH problem. We construct and adversary $\mathcal{B}$ against the X2DH problem. $\mathcal{B}$ receives $(E,\vec{\chi},T)$ as input where $\vec{\chi}=\left(g_1,g_1^{z},g_1^{r},g_1^{r'},g_2\right)$, and either $T=g_1^{zr}$ or $T=g_1^{zr'}$. $\mathcal{B}$ samples $y \leftarrow\!\$\ \mathbb{Z}_p$, and computes g_1^{y}, $(g_1^{r})^{y}$ and $(g_1^{r'})^{y}$. Note, these three elements along with $\vec{\chi}$ constitutes the required vector $\vec{x}$. $\mathcal{B}$ invokes $\mathcal{A}$ on $(E,\vec{x},T)$, and returns the response from $\mathcal{A}$. Clearly, $\mathcal{B}$ is correct whenever $\mathcal{A}$ is correct. From the distribution of the input to $\mathcal{B}$ and uniformity of y, the inputs to $\mathcal{A}$ are simulated perfectly to the problem condition. We conclude

$$\mathbf{Adv}^{\mathsf{X2DH}}_{\mathcal{B},\mathcal{G}}=\mathbf{Adv}^{\mathsf{AXDH}}_{\mathcal{A},\mathcal{G}}$$

Thus we conclude that for all PPTM $\mathcal{A}$, there exists PPTM $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathsf{AXDH}}_{\mathcal{A},\mathcal{G}} \le 2\mathbf{Adv}^{\mathsf{XDH}}_{\mathcal{B},\mathcal{G}}$$

H Thresholdized BTIB−KEM-2

In this section we present our second BTIB−KEM construction (BTIB−KEM−2) in detail. Upfront, we note that the BTIB−KEM−2 construction is defined with respect to an arbitrary but fixed constant ℓ_{max} such that $1 \le \ell \le \ell_{max}$, where ℓ_{max} is polynomial in the security parameter λ. This requirement is only necessary because of a minor technical detail in the semantic security proof, which requires the reduction to know an upper bound on ℓ upfront. This detail does not in any way impact the practicality of the scheme.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$
  - Generate a pairing ensemble $E=(\mathbb{G}_1,g_1,\mathbb{G}_2,g_2,\mathbb{G}_T,p,e)$.
  - Sample random group elements $u_1,u_2,v_1,v_2 \leftarrow\!\$\ \mathbb{G}_1$.
  - Additionally, we use a hash function H with output domain in $\mathbb{G}_1$.
  - Output $\mathsf{pp} \coloneqq (E,(u_1,u_2,v_1,v_2))$.
- $\mathsf{KGen}(1^\lambda,n,t,\ell) \to (\mathsf{pk},\mathsf{sk})$
  - Sample $\alpha,y,z \leftarrow\!\$\ \mathbb{Z}_p$.
  - Compute $X \leftarrow g_1^{\alpha}$, $Y \leftarrow g_2^{y}$, $Z \leftarrow g_2^{z}$ and $Q \leftarrow g_2^{\alpha yz}$.
  - Set $\mathsf{pk}=\{X,Y,Z,Q\}$
  - Additionally, we define the functions $\mathcal{L}_1,\mathcal{L}_2,\mathcal{R}_1,\mathcal{R}_2$ for $j \in [\ell]$, $\mathsf{ID} \in \mathbb{Z}_p$ as follows:

$$\mathcal{L}_1(j)=X^{j}\cdot u_1\ [=g_1^{\alpha j}\cdot u_1]$$
$$\mathcal{L}_2(\mathsf{ID})=X^{\mathsf{ID}}\cdot u_2$$
$$\mathcal{R}_1(j)=X^{j}\cdot v_1$$
$$\mathcal{R}_2(\mathsf{ID})=X^{\mathsf{ID}}\cdot v_2$$

    Note that they can be evaluated by anyone in possession of the public pk and the public parameters p.
  - Generate a (t,n)-sharing $\{\alpha_i\}_{i\in[n]}$ of α.
  - For every $(j,i) \in [\ell]\times[n]$:
    - Sample $r_{1,i}^{(j)},h_{1,i}^{(j)} \leftarrow\!\$\ \mathbb{Z}_p$.

* Compute the left key-shares as:

$$sk_{i,0}^{(j)} \leftarrow (H(j)^{\alpha_i z} \cdot \mathcal{L}_1(j)^{r_{1,i}^{(j)}}, Y^{r_{1,i}^{(j)}}, g_2^{r_{1,i}^{(j)}}, g_2^{z\alpha_i})$$

* Compute the right key-shares as

$$sk_{i,1}^{(j)} \leftarrow (H(j)^{\alpha_i y} \cdot \mathcal{R}_1(j)^{h_{1,i}^{(j)}}, Z^{h_{1,i}^{(j)}}, g_2^{h_{1,i}^{(j)}}, g_2^{y\alpha_i})$$

- Set $\mathsf{sk} \leftarrow \{(sk_{i,0}^{(j)}, sk_{i,1}^{(j)}\}_{j\in[\ell],i\in[n]}$
- Output pk, sk.

- $\mathsf{Enc}(pk, j, \mathsf{ID}) \rightarrow (k, c = (c_0, c_1))$
  - Sample $s, t_0, t_1 \leftarrow\!\!\$\ \mathbb{Z}_p$
  - Compute the key $k \leftarrow e(H(j)^s, Q)$ $[= e(H(j), g_2)^{\alpha yzs}]$.
  - Set the left ciphertext

$$c_0 \coloneqq (c_{0,1}, c_{0,2}, c_{0,3}, c_{0,4}, c_{0,5}, c_{0,6}) \leftarrow (Y^s \cdot g_2^{t_0}, \mathcal{L}_1(j)^s, \mathcal{L}_2(\mathsf{ID})^s, \mathcal{L}_1(j)^{t_0}, \mathcal{L}_2(\mathsf{ID})^{t_0}, H(j)^{t_0})$$

  - Set the right ciphertext

$$c_1 \coloneqq (c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, c_{1,5}, c_{1,6}) \leftarrow (Z^s \cdot g_2^{t_1}, \mathcal{R}_1(j)^s, \mathcal{R}_2(\mathsf{ID})^s, \mathcal{R}_1(j)^{t_1}, \mathcal{R}_2(\mathsf{ID})^{t_1}, H(j)^{t_1})$$

  - Assemble $c = (c_0, c_1)$ and output (k, c)
- $\mathsf{DIdk}([\mathsf{sk}_{i,b_{i,j}}^{(j)}]_{j\in[\ell]}, \mathsf{ID}) \rightarrow \mathsf{idk}_i$ The identity key derivation algorithm uses either the left or right secret key share per index j of party i to derive a left or right identity key share respectively. For left shares, the bit $b_{i,j}$ is 0, for right shares it is 1.
  - For all $j \in [\ell]$ do:
    * Let $\mathsf{sk}_i \leftarrow \mathsf{sk}_{i,b_{i,j}}^{(j)}$
    * If $b_{i,j} = 0$, then sample $r_{2,i}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute the left partial identity key as

$$(w_{j,i,1}, w_{j,i,2}, w_{j,i,3}, w_{j,i,4}, w_{j,i,5}, w_{j,i,6}) \leftarrow (\mathsf{sk}_{i,1} \cdot \mathcal{L}_2(\mathsf{ID})^{r_{2,i}^{(j)}}, \mathsf{sk}_{i,2}, Y^{r_{2,i}^{(j)}}, \mathsf{sk}_{i,3}, g_2^{r_{2,i}^{(j)}}, \mathsf{sk}_{i,4})$$

    * Otherwise, if $b_{i,j} = 1$, then sample $h_{2,i}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute the right partial identity key as

$$(w_{j,i,1}, w_{j,i,2}, w_{j,i,3}, w_{j,i,4}, w_{j,i,5}, w_{j,i,6}) \leftarrow (\mathsf{sk}_{i,1} \cdot \mathcal{R}_2(\mathsf{ID})^{h_{2,i}^{(j)}}, \mathsf{sk}_{i,2}, Z^{h_{2,i}^{(j)}}, \mathsf{sk}_{i,3}, g_2^{h_{2,i}^{(j)}}, \mathsf{sk}_{i,4})$$

    * Set $\mathsf{idk}_i^{(j)} \leftarrow (w_{j,i,1}, w_{j,i,2}, w_{j,i,3}, w_{j,i,4}, w_{j,i,5}, w_{j,i,6})$.
  - Output $\mathsf{idk}_i \leftarrow [(b_{i,j}, \mathsf{idk}_i^{(j)})]_{j\in[\ell]}$.
- $\mathsf{CombIdk}(S, [\mathsf{idk}_i]_{i\in S}) \rightarrow \mathsf{idk}$ This algorithm combines the identity key shares obtained through previous DIdk queries to the Servers in S.
  - Require that $S \subseteq [n]$, where $|S| \geq t$.
  - For all $j \in [\ell]$ do:
    * Denote by $S_0 \leftarrow \{i \in S \mid b_{i,j} = 0\}$ the set of all servers who have provided left identity key shares for index j.
    * Denote by $S_1 \leftarrow S \setminus S_0$ the set of all servers who provided right identity key shares of index j. Observe that, by construction, $S_0 \cup S_1 = S$ and $S_0 \cap S_1 = \emptyset$.
    * Compute the left identity key of position j:

$$L^{(j)} \coloneqq (L_1^{(j)}, \ldots, L_6^{(j)}),$$

      where $L_m^{(j)} \leftarrow \prod_{i\in S_0} w_{j,i,m}^{l_{i,S}(0)}$ for $m \in [1, 6]$. Here, we partially aggregate the left identity key shares derived by parties from S_0 through interpolation at position 0 among the set S.
    * Compute the right identity key for position j:

$$R^{(j)} \coloneqq (R_1^{(j)}, \ldots, R_6^{(j)}),$$

      where $R_m^{(j)} \leftarrow \prod_{i\in S_1} w_{j,i,m}^{l_{i,S}(0)}$ for $m \in [1, 6]$. We aggregate all right shares also by partial interpolation among S.
  - Return $\mathsf{idk} = (L^{(j)}, R^{(j)})_{j\in[\ell]}$.
- $\mathsf{Dec}(\mathsf{idk}, c = (c_0, c_1), j) \rightarrow k$
  - Compute

$$k_1 \leftarrow \frac{e(L_1^{(j)}, c_{0,1})}{\prod_{m=2}^{6} e(c_{0,i}, L_m^{(j)})},$$

$$k_2 \leftarrow \frac{e(R_1^{(j)}, c_{1,1})}{\prod_{m=2}^{6} e(c_{1,i}, R_m^{(j)})}.$$

  - Return $k \leftarrow k_1 \cdot k_2$.

H.1 Security

We prove semantic security and one-sided security by reduction to the following two assumptions.

Definition H.1 (ABDDH-2 Assumption). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the ABDDH-2 problem is hard for $\mathcal{G}$, if for all PPT adversaries it holds that

$$\left|\Pr\left[\mathcal{A}(E, \vec{x}, e(g_1, g_2)^{w\alpha yzs}) = 1\right] - \Pr\left[\mathcal{A}(E, \vec{x}, e(g_1, g_2)^{v}) = 1\right]\right|$$

is negligible in λ where $E \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$, $w, \alpha, y, z, s, v \leftarrow\!\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \left(g_1, g_1^w, g_1^{wz}, g_1^{wy}, g_1^\alpha, g_1^s, g_2, g_2^y, g_2^z, g_2^{wyz}, g_2^{ys}, g_2^{zs}, g_2^{\alpha y}, g_2^{\alpha z}, g_2^{\alpha yz}\right).$$

Definition H.2 (AXDH-2 Assumption). Let $\mathcal{G}$ be an algorithm to generate ensembles E. We say that the AXDH-2 problem is hard for $\mathcal{G}$, if for all PPT adversaries it holds that

$$\left|\Pr\left[\mathcal{A}(E, \vec{x}, (g_1^{s+\bar{v}_1 s}, g_1^{s+\bar{v}_2 s}, g_2^{zs+t_1})) = 1\right] - \Pr\left[\mathcal{A}(E, \vec{x}, (g_1^{s'+\bar{v}_1 s'}, g_1^{s'+\bar{v}_2 s'}, g_2^{zs'+t_1})) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where $E \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$, $w, y, z, s, s' t_0, t_1, \bar{u}_1, \bar{u}_2, \bar{v}_1, \bar{v}_2 \leftarrow\!\!\$\ \mathbb{Z}_p$ and

$$\vec{x} = \big(g_1, g_1^y, g_1^z, g_1^w, g_1^{wz}, g_1^{wt_1}, g_1^{wt_0}, g_1^{\bar{u}_1}, g_1^{\bar{v}_1}, g_1^{\bar{u}_2}, g_1^{\bar{v}_2}, g_1^{t_1+\bar{v}_1 t_1}, g_1^{t_0+\bar{u}_1 t_0}, g_1^{s+\bar{u}_1 s}, g_1^{t_1+\bar{v}_2 t_1}, g_1^{t_0+\bar{u}_2 t_0}, g_1^{s+\bar{u}_2 s}, g_2, g_2^y, g_2^z, g_2^{yz}, g_2^{ys+t_0}, e(g_1, g_2)^{wyzs}, e(g_1, g_2)^{wyzs'}\big)$$

Hardness of the Assumptions. Both assumptions can be seen as instances of the Uber problem for bilinear groups [8, 20], which has been shown to be hard in the (bilinear) generic group model (GGM) if there is no trivial way to compute the challenge from elements in $\vec{x}$ using group and pairing operations [8, 47]. Particularly for the more complex AXDH-2 assumption, this was verified by checking that all pairing results between the source-group challenge elements and elements from $\vec{x}$ do not appear as non-trivial linear combinations of pairings between elements from $\vec{x}$ and the challenge[16]. As this is the case for both of the above, the ABDDH-2 and AXDH-2 assumptions hold in the GGM.

H.1.1 Semantic Security.

Lemma H.3 (IND–SID–CPA security of BTIB–KEM–2). *Our* BTIB–KEM–2 *construction is* IND–SID–CPA *secure if the ABDDH-2 assumption (Definition H.1) holds in the underlying pairing ensemble.*

Proof. We prove IND–SID–CPA security of BTIB–KEM–2 by reduction to the ABDDH-2 assumption (Definition H.1). Let $\mathcal{A}$ be a PPT adversary against Game–IND–SID–CPA (Figure 5) such that

$$\Pr[\text{Game–IND–SID–CPA}_{\mathcal{A}}^{\text{BTIB–KEM–2}}(1^\lambda) = 1] \geq 1/2 + \varepsilon(\lambda)$$

for a non-negligible ε. We construct a PPT reduction $\mathcal{B}$ that uses $\mathcal{A}$ internally to break the ABDDH-2 assumption.

First, $\mathcal{B}$ receives the security parameter 1^λ as well as an ABDDH-2 instance

$$g_1, g_1^w, g_1^{wz}, g_1^{wy}, g_1^\alpha, g_1^s$$
$$g_2, g_2^y, g_2^z, g_2^{wyz}, g_2^{ys}, g_2^{zs}, , g_2^{\alpha y}, g_2^{\alpha z}, g_2^{\alpha yz}$$

with challenge T.

Let ℓ_{max} be the upper bound on ℓ for BTIB–KEM–2. First, the reduction $\mathcal{B}$ samples a random $j^* \in [\ell_{max}]$. We note here that at this point, the adversary $\mathcal{A}$ did not output ℓ yet, hence the BTIB–KEM–2 construction defines an upper bound ℓ_{max} such that $1 \leq \ell \leq \ell_{max}$. Then, the reduction $\mathcal{B}$ runs $\mathcal{A}_1$ to get ID^* and ℓ while simulating the random oracle as follows.

Simulating the random oracle H. For each query $\mathcal{H}(j)$, $\mathcal{B}$ simulates the random oracle H as follows: If $j = j^*$, then output the group element g_1^w from the assumption instance. Otherwise, sample a random $w_j \leftarrow\!\!\$\ \mathbb{Z}_p$ and output $g_1^{ww_j}$. $\mathcal{B}$ answers queries *consistently*, i.e. it saves the values and repeats answers for identical queries.

If $j^* \notin [\ell]$, then $\mathcal{B}$ immediately returns a random bit. The probability for this to happen can be bounded by $1/\ell_{\max}$.

Then, $\mathcal{B}$ prepares a simulation of the public parameters and KGen. For this, the challenge is embedded in the public key as $X \leftarrow g_1^\alpha$, $Y \leftarrow g_2^y$, $Z \leftarrow g_2^z$, and $Q \leftarrow g_2^{\alpha yz}$. Further, $\mathcal{B}$ samples $k_{0,1}, k_{0,2}, k_{1,1}, k_{1,2} \leftarrow\!\!\$\ \mathbb{Z}_p$ to set the public parameters

- $u_1 \leftarrow (g_1^\alpha)^{-j^*} \cdot g_1^{k_{0,1}}$
- $u_2 \leftarrow (g_1^\alpha)^{-\mathsf{ID}^*} \cdot g_1^{k_{0,2}}$
- $v_1 \leftarrow (g_1^\alpha)^{-j^*} \cdot g_1^{k_{1,1}}$
- $v_2 \leftarrow (g_1^\alpha)^{-\mathsf{ID}^*} \cdot g_1^{k_{1,2}}$

[16]We note that many possible combinations can be ruled out right away, as many target monomials do not appear at all in pairings among $\vec{x}$.

$\mathcal{B}$ sets $\mathsf{pk} \leftarrow (X, Y, Z, Q)$ and runs $\mathcal{A}_2(\mathsf{st}_1, \mathsf{pk})$ to obtain the set of corrupted parties C. W.l.o.g. we assume that $C = \{1, \ldots, t-1\}$. In the next step, the reduction needs to prepare a simulation of the secret key shares for corrupted parties in C as well as the challenge ciphertext and KEM key.

Simulating secret key shares. The reduction samples random $s_1, \ldots, s_{t-1} \leftarrow\!\!\$\ \mathbb{Z}_p$ to simulate a sharing of α, which is a standard simulation of Shamir's secret sharing. Hence, this Shamir simulation is with respect to a polynomal $p(i)$, where $p(0) = \alpha$ and $p(i) = s_i$ of $i \in C$. As the reduction only knows g_1^α, but not α, we can only interpolate p in the exponent. We will use this trick extensively in our simulation of identity-key derivation later. We can simulate $t-1$ secret shares for parties in C by sampling $r_{i,1}^{(j)}, h_{i,1}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_p$ for all $j \in [\ell], i \in C$ and setting the secret key shares as follows. For the left share, we set $\mathsf{sk}_{i,0}^{(j)} \leftarrow (e_{i,0}^{(j)}, Y^{r_{i,1}^{(j)}}, g_2^{r_{i,1}^{(j)}}, g_2^{zs_i})$, where

$$e_{i,0}^{(j)} \leftarrow (g_1^{wz})^{w_j s_i} \cdot (g_1^\alpha)^{j \cdot r_{i,1}^{(j)}} \cdot u_1^{r_{i,1}^{(j)}} \; [= H(j)^{zs_i} \cdot \mathcal{L}_1(j)^{r_{i,1}^{(j)}}].$$

For the right share, we set $\mathsf{sk}_{i,1}^{(j)} \leftarrow (e_{i,1}^{(j)}, Z^{h_{i,1}^{(j)}}, g_2^{h_{i,1}^{(j)}}, g_2^{ys_i})$, where

$$e_{i,1}^{(j)} \leftarrow (g_1^{wy})^{w_j s_i} \cdot (g_1^\alpha)^{j \cdot h_{i,1}^{(j)}} \cdot v_1^{h_{i,1}^{(j)}} \; [= H(j)^{ys_i} \cdot \mathcal{R}_1(j)^{r_{i,1}^{(j)}}].$$

To prepare the challenge key and ciphertext, the reduction samples $t_0, t_1 \in \mathbb{Z}_p$ and embeds the instance of the ABDDH-2 assumption and sets the key to the ABDDH-2 challenge $k \leftarrow T$. The ciphertext c_0 is simulated as follows.

- $Y^s \leftarrow g_2^{ys} \cdot g_2^{t_0}$
- $\mathcal{L}_1(j^*)^s = (g_1^{\alpha j^*} \cdot u_1)^s = (g_1^{\alpha j^*} \cdot g_1^{-\alpha j^* + k_{0,1}})^s \leftarrow (g_1^s)^{k_{0,1}}$
- $\mathcal{L}_2(\mathsf{ID}^*)^s \leftarrow (g_1^s)^{k_{0,2}}$
- $\mathcal{L}_1(j^*)^{t_0} \leftarrow g_1^{t_0 \cdot k_{0,1}}$
- $\mathcal{L}_2(\mathsf{ID}^*)^{t_0} \leftarrow g_1^{t_0 \cdot k_{0,2}}$
- $H(j^*)^{t_0} \leftarrow (g_1^w)^{t_0}$

and the right ciphertext c_1 is simulated similarly using the elements g_2^{zs} and g_1^s from the assumption.

Finally, $\mathcal{B}$ runs $b' \leftarrow\!\!\$\ \mathcal{A}_3^{\mathcal{O}^{\mathsf{DIdk}}}(\mathsf{st}_2, (c, k), \{\mathsf{sk}_i\}_{i \in C})$, while simulating the $\mathcal{O}^{\mathsf{DIdk}}$ oracle and returns the same bit b'.

Simulating $\mathcal{O}^{\mathsf{DIdk}}$. Given a query on (x, ID), the identity key shares from $x \in C$ can be simulated honestly from their secret shares. We next show how to simulate the oracle for $x \notin C$. The simulation returns $\perp$ if $\mathsf{ID} = \mathsf{ID}^*$. Here, we focus on how to simulate the left identity-key share. The simulation for the right identity-key share works in a similar manner, using the elements g_1^{wy}, g_2^z, and g_2^{wyz} from the assumption and sampling new random elements (the random elements used for the right side are independent from the ones used on the left side). Then:

- For $j \neq j^*$. Sample $r_{x,1}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_p$ and let in the following $l_i = l_{i,C \cup \{0\}}(x)$ be the Lagrange coefficient for interpolation at

target index x for index i among the set $C \cup \{0\}$. Compute

$$w'_{j,x,1} \leftarrow (g_1^{wz})^{\frac{-w_j k_{0,1} \cdot l_0}{j-j^*}} \cdot \left((g_1^{\alpha})^{j-j^*} \cdot g_1^{k_{0,1}}\right)^{r_{x,1}^{(j)}} \cdot \prod_{i=1}^{t-1} (g_1^{wz})^{w_j l_i s_i}$$

$$= g_1^{\frac{-w w_j z k_{0,1} \cdot l_0}{j-j^*}} \cdot g_1^{(\alpha j - \alpha j^* + k_{0,1}) \cdot r_{x,1}^{(j)}} \cdot \prod_{i=1}^{t-1} g_1^{w w_j z l_i s_i}$$

$$= g_1^{\frac{-w w_j z k_{0,1} \cdot l_0}{j-j^*}} \cdot \mathcal{L}_1(j)^{r_{x,1}^{(j)}} \cdot \prod_{i=1}^{t-1} g_1^{w w_j z l_i s_i}$$

$$= g_1^{w w_j z \alpha l_0} \cdot \prod_{i=1}^{t-1} g_1^{w w_j z l_i s_i} \cdot \mathcal{L}_1(j)^{r_{x,1}^{(j)} - \frac{w_j w z l_0}{j-j^*}}.$$

Further we set

$$w_{j,x,4} \leftarrow (g_1^{wz})^{w_j \frac{-l_0}{j-j^*}} \cdot g_1^{r_{x,1}^{(j)}} = g_1^{\frac{-w w_j z \cdot l_0}{j-j^*} + r_{x,1}^{(j)}}.$$

Note that these are valid parts of the identity key share. For this, consider $\tilde{r}_{x,1}^{(j)} = r_{x,1}^{(j)} - \frac{w w_j z l_0}{j-j^*}$. Then we get that

$$w'_{j,x,1} = g_1^{w w_j z \alpha l_0} \cdot \prod_{i=1}^{t-1} g_1^{w w_j z l_i s_i} \cdot \mathcal{L}_1(j)^{r_{x,1}^{(j)} - \frac{w_j w z l_0}{j-j^*}}$$

$$= g_1^{w w_j z \alpha_x} \cdot \mathcal{L}_1(j)^{\tilde{r}_{x,1}^{(j)}} [= H(j)^{z \alpha_x} \cdot \mathcal{L}_1(j)^{\tilde{r}_{x,1}^{(j)}}].$$

Also, it holds that $w_{j,x,4} = g_1^{\tilde{r}_{x,1}^{(j)}}$.

Next, we compute $w_{j,x,2} \leftarrow (g_2^y)^{r_{x,1}^{(j)}} \cdot (g_2^{wyz})^{w_j \frac{-l_0}{j-j^*}} = Y^{\tilde{r}_{x,1}^{(j)}}$ and $w_{j,x,6} \leftarrow (g_2^{\alpha z})^{l_0} \cdot \prod_{i=1}^{t-1} (g_2^z)^{s_i l_i} = g_2^{\alpha_x z}$. To finish, we compute the part of the second layer honestly, that is: we sample $r_{x,2}^{(j)}$ and compute $w_{j,x,1} \leftarrow w'_{j,x,1} \cdot \mathcal{L}_2(\mathsf{ID})^{r_{x,2}^{(j)}}$, $w_{j,x,3} \leftarrow Y^{r_{x,2}^{(j)}}$, and $w_{j,x,5} \leftarrow g_2^{r_{x,2}^{(j)}}$. Note that $(w_{j,x,1}, \ldots, w_{j,x,6})$ is a valid left decryption key share from party x for identity ID and position j.

- For $j = j^*$. The case is handled similarly. The only difference is that we now simulate the part of the first layer honestly and apply the previous trick to the second layer.

Analysis. Observe that, if $T = e(g_1, g_2)^{w \alpha yzs}$, then $\mathcal{B}$'s simulates $\mathsf{Game{-}IND{-}SID{-}CPA}$ for $b = 1$. If instead $T = e(g_1, g_2)^v$, $\mathcal{B}$'s simulation of k is uniformly random from $\mathcal{K}(\lambda)$, hence $\mathcal{B}$ simulates the game for $b = 0$. Bounding the loss of guessing $j^* \in [\ell]$ by $1/\ell_{max}$, we get that

$$\Pr[\mathsf{Game{-}IND{-}SID{-}CPA}_{\mathcal{A}}^{\mathsf{BTIB-KEM-2}}(1^\lambda) = 1] \leq \ell_{max} \cdot \mathbf{Adv}_{\mathcal{G},\mathcal{B}}^{\mathsf{AXDH-2}}$$

□

H.1.2 One-sided Security. Our result of one-sided security is captured by the following Lemma. We prove it by reduction to the AXDH-2 assumption (Definition H.2).

Lemma H.4. *For all PPT adversaries $\mathcal{A}$, there exists a PPT reduction $\mathcal{B}$ such that*

$$\mathsf{Game{-}OSS{-}ID}_{\mathcal{A}}^{\mathsf{BTIB-KEM-2}}(1^\lambda) \leq \ell_{max} \mathbf{Adv}_{\mathcal{G},\mathcal{B}}^{\mathsf{AXDH-2}}$$

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

Proof. The adversary $\mathcal{B}$, who is solving the AXDH-2 problem, receives a challenge T which is either $(g_1^{s+\bar{v}_1 s}, g_1^{s+\bar{v}_2 s}, g_2^{zs+t_1})$ or $(g_1^{s'+\bar{v}_1 s'}, g_1^{s'+\bar{v}_2 s'}, g_2^{zs'+t_1})$. It also receives the following group elements:

- From $\mathbb{G}_1$: $g_1, g_1^y, g_1^z, g_1^w, g_1^{wz}, g_1^{wt_1}, g_1^{wt_0}, g_1^{\bar{u}_1}, g_1^{\bar{v}_1}, g_1^{\bar{u}_2}, g_1^{\bar{v}_2}, g_1^{t_1+\bar{v}_1 t_1}, g_1^{t_0+\bar{u}_1 t_0}, g_1^{s+\bar{u}_1 s}, g_1^{t_1+\bar{v}_2 t_1}, g_1^{t_0+\bar{u}_2 t_0}, g_1^{s+\bar{u}_2 s}$.
- From $\mathbb{G}_2$: $g_2, g_2^y, g_2^z, g_2^{yz}, g_2^{ys+t_0}$.
- From $\mathbb{G}_T$: $e(g_1, g_2)^{wyzs}, e(g_1, g_2)^{wyzs'}$.

First, $\mathcal{B}$ guesses $j^* \leftarrow\!\!\$\ [\ell_{max}]$.

Simulating the random oracle H. For each query $H(j)$, $\mathcal{B}$ simulates the random oracle H as follows:

- If $j = j^*$, return g_1^w from the AXDH-2 instance.
- Otherwise, sample a random $w_j \leftarrow\!\!\$\ \mathbb{Z}_p$ and output $g_1^{w_j}$. $\mathcal{B}$ answers queries *consistently*, i.e. it saves the values and repeats answers for identical queries.
- We assume, $\mathcal{A}$ makes a hash query on j^*; if not, the reduction makes the query $H(j^*)$ itself.

$\mathcal{B}$ runs $\mathcal{A}$ to get $(\ell, u, d, \mathsf{ID}^*)$ and aborts if $u \neq j^*$.

Simulating Setup. The reduction $\mathcal{B}$ computes the public parameters using elements from the assumption. That is, it first samples $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$ and then sets $u_1 = (g_1^{\bar{u}_1})^{\alpha j^*}$, $u_2 = (g_1^{\bar{u}_2})^{\alpha \mathsf{ID}^*}$, $v_1 = (g_1^{\bar{v}_1})^{\alpha j^*}$, and $v_2 = (g_1^{\bar{v}_2})^{\alpha \mathsf{ID}^* \bar{v}_2}$. Hence the function $\mathcal{L}$ and $\mathcal{R}$ behave as follows:

$$\mathcal{L}_1(j) = g_1^{\alpha j} \cdot u_1 = g_1^{\alpha j} \cdot g_1^{\alpha j^* \bar{u}_1}$$
$$\mathcal{L}_2(\mathsf{ID}) = g_1^{\alpha \mathsf{ID}} \cdot u_2 = g_1^{\alpha \mathsf{ID}} \cdot g_1^{\alpha \mathsf{ID}^* \bar{u}_2}$$
$$\mathcal{R}_1(j) = g_1^{\alpha j} \cdot v_1 = g_1^{\alpha j} \cdot g_1^{\alpha j^* \bar{v}_1}$$
$$\mathcal{R}_2(\mathsf{ID}) = g_1^{\alpha \mathsf{ID}} \cdot v_2 = g_1^{\alpha \mathsf{ID}} \cdot g_1^{\alpha \mathsf{ID}^* \bar{v}_2}$$

We start with the case $d = 0$ first. In this case, the adversary is tasked to determine whether the given right side ciphertext c_1 fits with a correctly derived key and left ciphertext pair (k, c_0).

Simulating key generation. Given the above public parameters. the AXDH-2 instance, as well as α, we simulate key generation as follow:

- Generate (t, n)-Shamir shares $s_1, \ldots, s_n$ of α.
- For the public key, we set $X \leftarrow g_1^{\alpha}$, $Y \leftarrow g_2^y$, $Z \leftarrow g_2^z$, and $Q \leftarrow (g_2^{yz})^{\alpha}$.
- For $j \neq u$ the secret shares can be simulated since $H(j)^{zs_i} = (g_1^z)^{w_j s_i}$ where w_j and s_i are known to the reduction and g_1^z comes from the AXDH-2 instance. Then, $\mathcal{B}$ samples random $r_{i,1}^{(j)}, h_{i,1}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_p$ for all $(j, i) \in ([\ell] \setminus \{j^*\}) \times [n]$ and sets the left share as

$$sk_{i,0}^{(j)} \leftarrow \left((g_1^z)^{w_j s_i} \cdot \mathcal{L}_1(j)^{r_{i,1}^{(j)}}, (g_2^y)^{r_{i,1}^{(j)}}, g_2^{r_{i,1}^{(j)}}, (g_2^z)^{s_i}\right),$$

  and the right share as

$$sk_{i,1}^{(j)} \leftarrow \left((g_1^y)^{w_j s_i} \cdot \mathcal{R}_1(j)^{h_{i,1}^{(j)}}, (g_2^z)^{h_{i,1}^{(j)}}, g_2^{h_{i,1}^{(j)}}, (g_2^y)^{s_i}\right).$$

- For $j = u$ we need to simulate only the left shares. They can be simulated using the element g_1^{zw} from the AXDH-2 instance, since $H(u)^{zs_i} = (g_1^{zw})^{s_i}$. The adversary $\mathcal{B}$ samples

$r_{i,1}^{(j)}$ for every $i \in [n]$ and then the left shares are computed as:

$$sk_{i,0}^{(u)} \leftarrow \left((g_1^{zw})^{s_i} \cdot \mathcal{L}_1(u)^{r_{i,1}^{(u)}}, (g_2^{y})^{r_{i,1}^{(u)}}, g_2^{r_{i,1}^{(u)}}, (g_2^{z})^{s_i} \right)$$

Simulating the KEM key and ciphertext. $\mathcal{B}$ sets the KEM key $k_0 \leftarrow (e(g_1, g_2)^{wyzs})^{\alpha}$, where $(e(g_1, g_2)^{wyzs})$ is from the AXDH-2 instance. Conceptually, this means that $k_0 = e(H(j^*), g_2)^{\alpha yzs}$. The challenge ciphertext for (u, ID^*) is simulated as follows.

- The components of the left side, we simulate from the instance:
  - $c_{0,1} \leftarrow (g_1^{s+\bar{u}_1 s})^{\alpha j^*} [= (g_1^{\alpha j^*} \cdot g_1^{\bar{u}_1 \cdot \alpha j^*})^s = \mathcal{L}_1(j^*)^s]$
  - $c_{0,2} \leftarrow (g_1^{s+\bar{u}_2 s})^{\alpha \mathsf{ID}^*} [= (g_1^{\alpha \mathsf{ID}^*} \cdot g_1^{\bar{u}_1 \cdot \alpha \mathsf{ID}^*})^s = \mathcal{L}_2(\mathsf{ID}^*)^s]$
  - $c_{0,3} = g_2^{ys+t_0}$
  - $c_{0,4} = (g_1^{t_0+\bar{u}_1 t_0})^{\alpha j^*} [= \mathcal{L}_1(j^*)^{t_0}]$
  - $c_{0,5} = (g_1^{t_0+\bar{u}_2 t_0})^{\alpha \mathsf{ID}^*} [= \mathcal{L}_2(\mathsf{ID}^*)^{t_0}]$
  - $c_{0,6} = g_1^{wt_0} [= H(j^*)^{t_0}]$
- In order to set c_1, we use elements from the challenge. Denote the challenge by $T = (T_1, T_2, T_3)$, then:
  - $c_{1,1} \leftarrow (T_1)^{\alpha j^*}$, which is either $\mathcal{R}_1(j^*)^s$ or $\mathcal{R}_1(j^*)^{s'}$.
  - $c_{1,2} \leftarrow (T_2)^{\alpha \mathsf{ID}^*}$, which is either $\mathcal{R}_2(\mathsf{ID}^*)^s$ or $\mathcal{R}_2(\mathsf{ID}^*)^{s'}$.
  - $c_{1,3} = T_3$, which is either $g_2^{zs+t_1}$ or $g_2^{zs'+t_1}$.
  - $c_{1,4} = (g_1^{t_1+\bar{u}_1 t_1})^{\alpha j^*} [= \mathcal{R}_1(j^*)^{t_1}]$
  - $c_{1,5} = (g_1^{t_1+\bar{u}_2 t_1})^{\alpha \mathsf{ID}^*} [= \mathcal{R}_2(\mathsf{ID}^*)^{t_1}]$
  - $c_{1,6} = g_1^{wt_1} [= H(j^*)^{t_1}]$

The adversary $\mathcal{B}$ passes k_0, c_0, c_1 together with the secret shares to $\mathcal{A}$. In the end the reduction $\mathcal{B}$ outputs what $\mathcal{A}$ outputs.

Analysis. Observe that c_0 is a valid ciphertext that is consistent with k_0. Furthermore, when $T = (g_1^{s+\bar{u}_1 s}, g_1^{s+\bar{u}_2 s}, g_2^{zs+t_1})$, then $\mathcal{B}$ simulates the case $b = 1$ as c_1 is consistent with c_0 and k_0. On the other hand, when $T = (g_1^{s'+\bar{u}_1 s'}, g_1^{s'+\bar{u}_2 s'}, g_2^{zs'+t_1})$, $\mathcal{B}$ simulates the case $b = 0$ since c_1 is not consistent with c_0 and k_0. Hence, $\mathcal{B}$ simulates the one-sided security game for $\mathcal{A}$ for the case $d = 0$.

The case $d = 1$ is handled symmetrically, except that now we set $k_0 = e(H(j^*), g_2)^{\alpha yzs'}$ and now we simulate c_0 using the challenge T (instead of c_1). Also, we swap y and z. Hence, when $T = (g_1^{s'+\bar{u}_1 s'}, g_1^{s'+\bar{u}_2 s'}, g_2^{zs'+t_1})$, it holds that c_0 is consistent with k_0 and not with c_1, thus $\mathcal{B}$ simulates $b = 0$. On the other hand, when $T = (g_1^{s+\bar{u}_1 s}, g_1^{s+\bar{u}_2 s}, g_2^{zs+t_1})$, then c_0 is consistent with c_1 but not with k_0, thus, $\mathcal{B}$ simulates $b = 1$. □

I Computational Efficiency

We present the concrete computational efficiency of CCA-secure TT−KEM, when instantiated with our two constructions in Figure 17. We conclude that for both constructions the computational efficiency is sufficient for practical applications. To see this, note that on modern curves that support pairings such as BLS12-381, even the most expensive operations (that is the pairing itself) are computable in less than 1 millisecond on consumer hardware[17].

[17]These results were obtained on an AMD RYZEN 5800x CPU based on the https://github.com/kilic/bls12-381 implementation.

Algo	With BTIB−KEM−1	With BTIB−KEM−2
Enc	$\Sigma.\mathsf{KGen} + \Sigma.\mathsf{Sign} + \mathcal{H}$ $+3\mathsf{Exp}_{\mathbb{G}_1} + 1\mathsf{Pair}$	$\Sigma.\mathsf{KGen} + \Sigma.\mathsf{Sign} + 2\mathcal{H}+$ $4\mathsf{Mul}_{\mathbb{G}_1} + 13\mathsf{Exp}_{\mathbb{G}_1}+$ $2\mathsf{Mul}_{\mathbb{G}_2} + 4\mathsf{Exp}_{\mathbb{G}_2} + 1\mathsf{Pair}$
PDec	$\Sigma.\mathsf{Verify} + \mathcal{H} + 1\mathsf{Exp}_{\mathbb{G}_2}$	$\Sigma.\mathsf{Verify} + 2\mathcal{H}$ $2\mathsf{Mul}_{\mathbb{G}_1} + 2\mathsf{Exp}_{\mathbb{G}_1} + 2\mathsf{Exp}_{\mathbb{G}_2}$
Comb	$(t-2)\mathsf{Mul}_{\mathbb{G}_2} + t\mathsf{Exp}_{\mathbb{G}_2}$ $2\mathsf{Pair} + 1\mathsf{Mul}_{\mathbb{G}_T}$	$(t-2)\mathsf{Mul}_{\mathbb{G}_1} + t\mathsf{Exp}_{\mathbb{G}_1}+$ $5(t-2)\mathsf{Mul}_{\mathbb{G}_2} + 5t\mathsf{Exp}_{\mathbb{G}_2}+$ $12\mathsf{Pair} + 11\mathsf{Mul}_{\mathbb{G}_T} + 1\mathsf{Exp}_{\mathbb{G}_T}$

Figure 17: Summary of the computational efficiency of important CCA-secure TT−KEM operations, when instantiating with BTIB−KEM−1 or BTIB−KEM−2 and a signature scheme Σ as building blocks.

J Consistency

J.1 Consistency Definition for TT−KEM

In the consistency experiment Game−Consist (Figure 18) the adversary obtains all secret key shares for all share holders. Then, the adversary outputs a ciphertext c with the attached proof π along with two sets $S, S' \in [n]$ that each contain at least t parties. Furthermore, the adversary outputs partial decryptions (shares) $\{d_i\}_{i \in S}$ and $\{d'_i\}_{i \in S'}$ on behalf of the parties in S and S'. The game verifies the proof as well as the partial decryption shares. The proof is verified using the Verify algorithm, which already exists for TT−KEM according to definition 3.1. We add an algorithm PVerify to verify partial decryptions. If all verifications pass, the game combines the decryption shares of S and the decryption shares of S'. The adversary wins, if the resulting KEM-keys are different or either decryption fails (returns $\perp$). Observe that without the proof π, the adversary can win the game in TT−KEM, which is based on BT−KEM, as the adversary can output a ciphertext with mismatching left and right parts.

$\mathsf{Game-Consist}_{\mathcal{A}}(1^\lambda)$

$(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{tk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda, n, t, 1^{1/\varepsilon(\lambda)})$

$(c, \pi), S, S', \{d_i\}_{i \in S}, \{d'_i\}_{i \in S'} \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda, \mathsf{pk}, \{\mathsf{sk}_i\}_{[n]})$

if $\mathsf{Verify}(\mathsf{pk}, c, \pi) = 0$ **then return** 0

if $S \not\subseteq [n] \vee |S| \leq t$ **then return** 0

if $S' \not\subseteq [n] \vee |S'| \leq t$ **then return** 0

if $\exists i \in S$: $\mathsf{PVerify}(\mathsf{pk}, c, d_i) = 0$ **then return** 0

if $\exists i \in S'$: $\mathsf{PVerify}(\mathsf{pk}, c, d'_i) = 0$ **then return** 0

$k \leftarrow \mathsf{Comb}(S, \{d_i\}_{i \in S}, c)$

$k' \leftarrow \mathsf{Comb}(S', \{d'_i\}_{i \in S'}, c)$

if $k \neq k' \vee k = \perp \vee k' = \perp$ **then return** 1

return 0

Figure 18: Consistency game for TT−KEM.

Definition J.1 (Consistency of Traceable Threshold KEM). We say that a TT−KEM protocol achieves consistency, if for all PPT adversaries $\mathcal{A} \coloneqq (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ there exists a negligible function $\mathsf{negl}(\lambda)$

such that

$$\Pr\left[\mathsf{Game{-}Consist}^{\mathsf{TT-KEM}}_{\mathcal{A}}(1^\lambda)=1\right] \le \mathsf{negl}(\lambda),$$

where Game−Consist is defined in Figure 18.

J.1.1 Verification of Partial Decryption. Given a ciphertext c including vk_e the partial decryption PDec boils down to deriving the partial identity key on the identity $\mathsf{ID} = \mathsf{vk}_e$ for index j. Looking at the BTIB−KEM construction (Section 5), the partial identity key derivation is computed as $d_i = H(j, \mathsf{vk}_e)^{\mathsf{sk}_i^{(j)}}$, which is basically a BLS-signature [13] on message (j, vk_e). Hence the correctness of partial decryptions can be easily verified through the BLS equation $e(g_1, d_i) = e(g_1^{sk_i^{(j)}}, H(j, \mathsf{vk}_e))$, where $g_1^{\mathsf{sk}_i^{(j)}}$ is part of the public key. We implement this check by letting the PVerify algorithm output 1, if the above equation holds and 0 otherwise.

J.2 Consistency of the modified TT−KEM

Theorem J.2 (Consistency of the modified TT−KEM). *Let* M−TT−KEM *be the modified* TT−KEM*-scheme that is described here and in Section 7.* M−TT−KEM *is consistent, if the proof system* Π *achieves knowledge-soundness and the DLog assumption holds in* $\mathbb{G}_1$.

Proof of Theorem J.2. We prove consistency in a series of game-hops. First, we replace the decryption shares of the adversary by honestly generated ones, then we extract the witness ω for the proof that is attached to the adversary's ciphertext and finally, we exclude the possibility that the adversary has generated the proof using the trapdoor. In the end, we show that both decryptions must yield the same key $k \neq \perp$ by correctness of TT−KEM.

Let Game_0 be the consistency game $\mathsf{Game{-}Consist}^{\mathsf{M-TT-KEM}}$ as displayed in Figure 18.

$\underline{\mathsf{Game}_1}$: In this game-hop, we generate honest decryption shares on behalf of the paries in S and S' and use the honestly generated shares instead of those provided by the adversary when reconstructing k and k'.

Lemma J.3. $\mathsf{Game}_0^{\mathsf{M-TT-KEM}}$ *and* $\mathsf{Game}_1^{\mathsf{M-TT-KEM}}$ *are equivalent, i.e. for all adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{Game}_{0,\mathcal{A}}^{\mathsf{M-TT-KEM}}(1^\lambda) = 1] = \Pr[\mathsf{Game}_{1,\mathcal{A}}^{\mathsf{M-TT-KEM}}(1^\lambda) = 1]$

Proof of Lemma J.3. Due to the bilinearity of the pairing operation e, the honestly generated decryption shares must be equal to the decryption shares provided by the adversary, if the share verification PVerify passes (i.e. if $e(g_1, d_i) = e(g_1^{\mathsf{sk}_i^{(j)}}, H(j, \mathsf{vk}_e))$). □

$\underline{\mathsf{Game}_2}$: Next, we require the adversary to send a witness ω along with the ciphertext c and proof π and abort, if $(\chi, \omega) \notin \mathsf{R}$ for $\chi = (\mathsf{pk}, c_0, c_1, j, \mathsf{vk}_e)$. In particular, Game_2 aborts, if $g_1^\omega \neq Q$ and either $c_0 \neq Y_j^\omega$ or $c_1 \neq Z_j^\omega$.

Lemma J.4. *If the proof system* Π *is knowledge-sound (Definition C.3), then for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *and PPT adversary* $\mathcal{B}$ *such that*

$$\Pr[\mathsf{Game}_{2,\mathcal{B}}^{\mathsf{M-TT-KEM}}(1^\lambda)=1] \ge \Pr[\mathsf{Game}_{1,\mathcal{A}}^{\mathsf{M-TT-KEM}}(1^\lambda)=1]| - \mathsf{negl}(\lambda)$$

Proof of Lemma J.4. Let $\mathcal{A}$ be a PPT adversary such that

$$\Pr[\mathsf{Game}_{1,\mathcal{A}}^{\mathsf{M-TT-KEM}}(1^\lambda)=1] = \varepsilon(\lambda)$$

for a non-negligible $\varepsilon(\lambda)$. We construct an adversary $\mathcal{B}$ that internally uses $\mathcal{A}$ as well as the extractor Ext to win Game_2. $\mathcal{B}$ runs $\mathcal{A}$ to receive the tuple (c, π) as well as the partial decryptions and then runs $\mathsf{Ext}^{\mathcal{A}(\cdot)}(\chi, \pi)$ to obtain a witness ω. $\mathcal{B}$ then returns c, π, the partial decryptions and the witness ω to Game_2. Note that due to the knowledge-soundness of Π, $\mathsf{Verify}(\mathsf{pk}, c, \pi) = 1$ but $(\chi, \omega) \notin \mathsf{R}$ can only happen with negligible probability, hence $\Pr[\mathsf{Game}_{2,\mathcal{B}}^{\mathsf{M-TT-KEM}}(1^\lambda)=1] \ge \varepsilon - \mathsf{negl}(\lambda)$. □

$\underline{\mathsf{Game}_3}$: In the next step, we add an additional abort condition. In particular, we abort if $g_1^\omega = Q$.

Lemma J.5. *If the DLog problem holds in* $\mathbb{G}_1$ *then both games are computationally indistinguishable*

$$\mathsf{Game}_2^{\mathsf{M-TT-KEM}} \approx_c \mathsf{Game}_3^{\mathsf{M-TT-KEM}}.$$

Proof of Lemma J.5. Let $\mathcal{A}$ be a PPT adversary with a non-negligible distinguishing advantage $\varepsilon(\lambda)$. Clearly, $\mathcal{A}$ can only distinguish the two games by triggering the additional abort condition, thus providing ω such that $g_1^\omega = Q$. We construct a PPT reduction $\mathcal{B}$ that uses $\mathcal{A}$ to break the DLog assumption with probability $\varepsilon(\lambda)$. Initially, $\mathcal{B}$ receives the group setup as well as a DLog challenge T. $\mathcal{B}$ sets $Q = T$ in the public key and runs $\mathcal{A}$ and returns ω. Whenever $\mathcal{A}$ triggers the additional abort, then $g_1^\omega = Q = T$, hence $\mathcal{B}$ breaks the DLog assumption. □

To close the proof, we observe that no adversary exists that can win Game_3. In particular it holds that $c_0 = Y_j^\omega$ and $c_1 = Z_j^\omega$. Following the correctness of TT−KEM, the Comb operations for both S and S' yield $k = e(g_1, g_2)^{\alpha_j y_j z_j \omega}$. □