Context-Dependent Threshold Decryption and its Applications

Dan Boneh[1], Benedikt Bünz[2,4], Kartik Nayak[3,4], Lior Rotem[5], and Victor Shoup[6]

[1]Stanford University
[2]N.Y.U.
[3]Duke University
[4]Espresso Systems
[5]The Hebrew University
[6]Offchain Labs

September 9, 2025

Abstract

In a threshold decryption system a secret key is split across a number of parties so that any threshold of them can decrypt a given ciphertext. We introduce a new concept in threshold decryption called a *decryption context*, which is an additional argument that is used during decryption. The context ensures that decryption shares that are generated for a ciphertext using different contexts are isolated from each other and cannot be jointly used to decrypt the ciphertext. For example, suppose the decryption threshold is t. Further, suppose that less than t decryption shares are generated for a ciphertext c under one context, and less than t decryption shares are generated for c under a different context. Then this set of shares is insufficient to decrypt c even if the total number of shares exceeds t. This new concept has several important applications, most notably for implementing an encrypted mempool in a consensus protocol. We give two CCA-secure threshold decryption constructions that support context. One is based on ElGamal encryption, and the other is generic showing how to add context to any CCA-secure threshold decryption system without changing the encryption algorithm.

Contents

1 Introduction **1**

2 Preliminaries **6**
2.1 Threshold decryption 6
2.1.1 Robustness. 7
2.1.2 Security. 10

3 A construction **12**

4 The linear one-more Diffie-Hellman assumption **15**

5 Security analysis of $\mathsf{E}_{\mathrm{htdh1}}$ **16**

6 A generic construction **20**
6.1 The construction 21
6.2 Robustness 23
6.3 Security 26

7 Applications **28**
7.1 Encrypted atomic broadcast 28
7.1.1 A simple framework for a general class of atomic broadcast protocols. 29
7.1.2 Piggybacking decryption shares. 30
7.2 Encrypted auction systems 31

8 Stronger notions of security **32**
8.1 Adaptive corruptions 32
8.2 Simulation CCA security 33

9 Conclusions and future work **36**

References **37**

A A Threshold IBE Scheme **41**

1 Introduction

Before talking about threshold decryption and the various security notions we will introduce, it is best to review threshold signature schemes, where similar notions have been more extensively studied.

In a threshold signature scheme, we have N parties, each of which holds a signing key. We restrict ourselves to schemes where the signing protocol is non-interactive. In such a scheme, users make signing requests to individual parties, and in response to a request to sign a message m, a party will give a **signature share** on m back to the user (without interacting with the other parties). Any t such signature shares on a message m may be combined to compute a signature on m. Here, t is the **reconstruction threshold** of the scheme.

The traditional notion of threshold security captures a form of security against chosen message attack (or CMA security). This notion of security is defined in terms of a chosen message attack game in which the adversary can ask for signature shares from parties on various messages of its choice. The adversary may also corrupt up to $t-1$ parties (either statically or adaptively) to obtain their signing keys. Security means that it is infeasible for an adversary to forge a signature on a message that is "protected". Here, a message m is said to be "protected" if no honest (i.e., non-corrupt) party generated a signature share for m. Let us call this **low-threshold CMA security**.

A stronger notion of security, which we shall call **high-threshold CMA security**, has been studied in the literature under different names [52, 5, 4]. This notion of security is defined by an attack game that is exactly the same as above, except that the definition of a "protected" message is broadened to include a larger class of messages. Namely, a message m is "protected" if $|\boldsymbol{L} \cup \boldsymbol{S}_m| < t$, where $\boldsymbol{L}$ is the set of corrupt parties and $\boldsymbol{S}_m$ is the set of honest parties that have generated a signature share for m.

High-threshold CMA security is quite natural, as it mimics the security property enjoyed by the "trivial" threshold scheme in which a "threshold signature" is just a collection of t regular signatures. It is also needed in the analysis of many protocols, such as consensus protocols, that rely on a "quorum intersection argument" (see, e.g., [20]). In the static corruption model, proving high-threshold CMA security typically requires protocol modifications and/or stronger assumptions.

The notion of high-threshold CMA security under static corruptions was introduced in [52], motivated by the need for this property in [20]. That paper also presented and analyzed a threshold signature scheme satisfying this property based on RSA. More recently, the paper [5, 4] reintroduced this same security property and showed that the standard BLS threshold signature scheme [14, 7] satisfies this property under a certain kind of "one-more Diffie-Hellman" assumption (this same result for BLS signatures was also proved in [42]).

It is not hard to show that under adaptive corruptions, low and high threshold CMA security are equivalent.[1] Under static corruptions such an equivalence is not known to be true, except when N is very small (see Theorem 1 in [4]). So the notion of high-threshold CMA security is mainly of interest only in the static corruption model.

Throughout this paper, we treat key generation as a trusted, centralized process. In a real system, key generation will typically be implemented by a distributed key generation (DKG) protocol

[1] Proof sketch. Convert a "high" attack into a "low" attack" as follows: guess which one of the messages m submitted as a signing request will be chosen by the adversary in its forgery; convert each signing request to a party P_ℓ on m to a corruption of P_ℓ.

that emulates this process. In such a system, the DKG protocol, as well as the protocols that use the signature scheme, may well withstand fewer than $t-1$ corruptions. For example, in the asynchronous setting, we typically have $N = 3f + 1$, where at most f parties may be corrupted, and $t = 2f + 1$. Nevertheless, the signature scheme itself retains its unforgeability property even if any $t - 1$ signing keys are leaked. In particular, the above high/low equivalence under adaptive corruptions (which requires that the threshold signature scheme withstand $t - 1$ corruptions) continues to hold.

Threshold decryption. Now we turn to threshold decryption. In a threshold decryption scheme, we have N parties, each of which holds a decryption key. We restrict ourselves to schemes where the decryption protocol is non-interactive. In such a scheme, users make decryption requests to individual parties, and in response to a request to decrypt a ciphertext c, a party will give a **decryption share** of c back to the user (without interacting with the other parties). Any t such decryption shares on a ciphertext c may be combined to compute the plaintext encrypted by c.

One can define corresponding notions of low and high threshold security for a threshold decryption scheme. Both of these notions capture a form of security against chosen ciphertext attack (CCA security). These notions of security are defined in terms of a chosen ciphertext attack game in which the adversary can ask for decryption shares from parties on various ciphertexts of its choice. The adversary may also corrupt up to $t - 1$ parties (either statically or adaptively) to obtain their decryption keys.

Low-threshold CCA security says that a ciphertext c remains "protected" (i.e, the adversary learns nothing about the message that c encrypts) if no honest party generates a decryption share for c. This is essentially the definition of security in [54] (who only considered static corruptions).

High-threshold CCA security says that a ciphertext c remains "protected" if $|\boldsymbol{L} \cup \boldsymbol{D}_c| < t$, where $\boldsymbol{L}$ is the set of corrupt parties and $\boldsymbol{D}_c$ is the set of honest parties that have generated a decryption share for c.

Note that [54] also introduces the notion of "associated data" (or a "label") that is an input to the encryption algorithm. Such "associated data" is public information that can be used to further protect a ciphertext, for example, by specifying a "decryption policy" for the ciphertext that can be used to protect against a rogue decryption request. Indeed, associated data is essential for most applications of threshold decryption.[2] For the purposes of this immediate discussion on defining security, we can assume that the encryption algorithm embeds the "associated data" in the ciphertext c itself. We will make this more explicit when we present formal definitions in Section 2.

We show in Theorem 4 that, under adaptive corruptions, low and high threshold security are equivalent. Under static corruptions such an equivalence is not known to be true, except when N is very small.

Threshold decryption with decryption context. While the notion of high-threshold CCA security seems very natural, there is very little discussion of it in the literature. Perhaps one reason for the lack of research in this area is that this type of security does not *by itself* seem to enable any compelling applications. However, we propose an enhanced version of this notion that *does*.

[2] In the non-threshold setting, associated data can be encrypted along with the plaintext, or processed by a symmetric cipher in a hybrid construction. But in the threshold setting, this simply does not work, as by the time the associated data is checked, the plaintext is already revealed.

This enhanced version of high-threshold CCA security works as follows. In addition to a ciphertext c and decryption key share, the decryption algorithm (i.e., the algorithm that generates a decryption share) takes as input an application-specific **decryption context** dc. Security now means that a ciphertext c remains "protected" provided $|\boldsymbol{L} \cup \boldsymbol{D}_c^{(dc)}| < t$, for all decryption contexts dc. Here, $\boldsymbol{D}_c^{(dc)}$ is the set of honest parties who have generated a decryption share for c using decryption context dc. We refer to a threshold decryption scheme that supports a decryption context as a **context-dependent threshold decryption**. Traditional threshold decryption schemes are **context-free**.

To see an example, let $\boldsymbol{L} = \emptyset$ and $t \geq 2$, and let c be a valid ciphertext. Suppose that some $t-1$ parties provide decryption shares for c using a decryption context dc_1. Separately, another set of $t-1$ parties provide decryption shares for c using a different decryption context dc_2. Now, despite the presence of $2t-2 \geq t$ shares, it is not possible to decrypt c. This is because no set of t parties decrypted with the same context. In contrast, in a context-free threshold decryption system, these $2t-2$ shares would be more than enough to decrypt c. As we will see, the ability to isolate decryption shares by their context is required for security in many applications.

We stress that the notions of a *decryption context* (which is selected at the time of decryption and given as input to the decryption algorithm) and *associated data* (which is selected at the time of encryption and given as input to both encryption and decryption algorithms) are unrelated notions.

Applications. An important application for context-dependent threshold decryption is an encrypted mempool in decentralized systems [50, 3]. In many existing blockchains, transactions are submitted in the clear and they are publicly visible in the mempool before they are committed on chain. This leads to the MEV problem [32] where transactions can be front-run. One mitigation used in practice [45, 56, 39] is an encrypted mempool: transactions are submitted encrypted to a public key of a threshold decryption scheme. Once a block has enough votes to be added to the blockchain, a quorum of parties publish decryption shares for the transactions in the block. This protects transactions in the mempool, but delays block finalization due to the added round of communication for collecting decryption shares. This overhead can be eliminated by asking honest validators to publish a decryption share concurrently with voting for a block. This way, if the decryption threshold and the voting threshold are set to the same value t, then once a block has enough votes to be committed on chain, all of the transactions in it have enough decryption shares to be decrypted. But this is insecure. The problem is that one set of honest validators may vote for a block B_1 and another set of honest validators may vote for a block B_2. As a result, a transaction in the intersection of B_1 and B_2 might accumulate t public decryptions shares, even though neither block has t votes. This transaction will be decrypted before it is committed on chain, which violates the main goal of an encrypted mempool.

Context-dependent threshold decryption completely solves this problem: honest validators set the decryption context to be the hash of the block that they are voting for. This prevents mixing decryption shares for a transaction in B_1 with decryption shares for the same transaction in B_2. As we will see in Section 7, it ensures that a transaction can only be decrypted if it is in a block that is guaranteed to be committed on chain. We also show in Section 7.2 how a decryption context is needed in a sealed-bid auction system.

To further motivate the need for a decryption context, let us consider another simple application: Bob wants to build a *dead man's switch* for his password. To do so, Bob encrypts his password

under some public key and posts the resulting ciphertext publicly. The corresponding secret key is shared among N trustees with a reconstruction threshold of t. Once a day Bob sends a ping to all N trustees. Every trustee that receives a ping does nothing that day. However, if a trustee fails to receive a ping on a certain day, it sends a decryption share for Bob's ciphertext to his heirs. This way, when Bob becomes incapacitated, and does not send any pings, the trustees will release their decryption shares, and Bob's password will be revealed to his heirs. Importantly, the password should not be revealed as long as every day more than $N - t$ trustees receive the ping.

While this system may seem like a natural design, it is horribly insecure. If on Monday only $t/2$ trustees do not receive the ping, then $t/2$ decryption shares are released. If on Wednesday a different set of $t/2$ trustees do not receive the ping, then another $t/2$ decryption shares are released. Now the heirs have a total of t shares and can decrypt the password, even though, on every day more than $N - t$ trustees received the ping. This violates the goals of the system.

A decryption context is an easy solution to this problem. The trustees can use the current day's date as the decryption context when publishing a decryption share. This way the password is revealed *only if* t or more parties do not receive the ping on a single day. In particular, the set of t shares released in the preceding paragraph are insufficient to decrypt Bob's ciphertext — there is no set of t shares that were generated with the same decryption context.

More generally, a decryption context chosen at the time of decryption ensures that at least t parties must agree to decrypt the provided ciphertext *under the specified context* — otherwise, no information about the encrypted message is revealed. It is a cheap and simple way to ensure consensus among the parties during decryption.

Our results. We begin in Section 2 by giving precise definitions for secure context-dependent threshold decryption, both high-threshold and low-threshold.

In Section 3 we construct our first context-dependent high-threshold decryption system. The system is a variant of the ElGamal-based threshold scheme called TDH1 from [54]. We call this new scheme $\mathsf{E}_{\text{htdh1}}$. The encryption algorithm of $\mathsf{E}_{\text{htdh1}}$ is essentially identical to that of TDH1 — the only difference is that we use a more general symmetric encryption algorithm. The decryption and combiner algorithms are then modified to incorporate a decryption context. We prove the security of this system (in the random oracle model) in Section 5 using a falsifiable complexity assumption we call the *linear one-more Diffie-Hellman*, or LOMDH, which is essentially the same assumption that has been used to analyze high-threshold BLS signatures in [5, 42]. We note that it does not seem possible to prove that the original TDH1 scheme is itself context-free high-threshold secure under the LOMDH assumption — our modifications in the decryption and combiner algorithms seem essential to achieve any type of high-threshold security under this assumption.

We also show that the LOMDH assumption is implied by the decisional Diffie-Hellman (DDH) assumption. It follows that $\mathsf{E}_{\text{htdh1}}$ is high-threshold CCA secure under the DDH (in the random oracle model). However, LOMDH is a computational assumption, and so $\mathsf{E}_{\text{htdh1}}$ may also be secure in groups where the DDH assumption is false.

In Section 6 we construct a second context-dependent high-threshold decryption system. This construction is generic: it compiles any context-free low-threshold decryption system into one that is context-dependent and high-threshold secure. An important property of this compilation is that it does not modify the key generation or encryption algorithms of the underlying threshold scheme. This is particularly useful when the trustees want to add a decryption context to a legacy encryption system, without modifying how clients encrypt their data. The key ingredient that

makes this possible is threshold identity-based encryption [10].

In Section 7 we present two applications for context-dependent high-threshold decryption. One to encrypted atomic broadcast (an encrypted mempool) and the other to encrypted sealed-bid auctions. These primarily serve as examples where a decryption context enhances and simplifies the design of the system.

Our definitions and constructions only deal with static corruptions, with security being defined using a game-based definition. In Section 8 we present two stronger definitions of security. The first strengthens our core definition from Section 2 by allowing for adaptive corruptions. We leave it as future work to prove security of our schemes, or other schemes, under adaptive corruptions. Second, we present a simulation-based security definition that is compatible with the universal composability framework [22], and discuss the security of our constructions in that setting.

We conclude with a number of directions for future work, in particular, extending our constructions to support the recent notion of threshold decryption with silent setup [41] and batch decryption [28].

Additional related work. The basic notion of threshold decryption dates back to Desmedt [35, 36] and De Santis et al. [34]. Since then, many works constructed threshold decryption schemes from various assumptions and with different security guarantees. For conciseness, we focus here on CCA-secure threshold decryption schemes.

Shoup and Gennaro [54] were the first to construct a CCA-secure threshold decryption scheme, relying on the computational Diffie-Hellman assumption in the random oracle model. Subsequently, Fouque and Pointcheval [40] thresholdized the Naor-Yung paradigm for CCA security, by considering its instantiation based on Paillier encryption [49]. Canetti and Goldwasser [23] achieved CCA security in the standard model, by thresholdizing the Cramer-Shoup encryption scheme [31]. The latter construction relies on correlated randomness among decrypting parties, which requires either interactive decryption or pre-storing this randomness, bounding the number of possible decryptions. This restriction was lifted by Cramer, Damgård, and Ishai [30], but their solution is only efficient when N is small. Later on, Boneh et al. [13] constructed the first distributed pseudorandom functions without random oracles, which can be used to derandomize the Canetti-Goldwasser construction for any number of parties. Boyen, Mei, and Waters [18] and Boneh, Boyen, and Halevi [8] were the first to construct a CCA-secure threshold decryption scheme in the standard model, building on the IBE-to-CCA paradigm of Canetti, Halevi, and Katz [24]. Many subsequent works constructed CCA-secure threshold decryption schemes from various assumptions; see, for example [38, 57, 47, 11, 37] and the references therein for a non-exhaustive account.

Most threshold decryption schemes, including the ones mentioned above, are proven secure against static corruptions. There are, however, a few works constructing CCA-secure threshold decryption schemes that remain secure even against an adversary that can adaptively obtain secret keys. Jarecki and Lysyanskaya [44] and Abe and Fehr [1] presented adaptively-secure variants of the Canetti-Goldwasser threshold decryption scheme. However, both schemes inherent the limitations of the Canetti-Goldwasser scheme, requiring either interactive decryption or only supporting a bounded number of decryptions. Libert and Yung [48, 47] and Libert et al. [46] gave the first adaptively-secure non-interactive construction in bilinear groups. Recently, Devevey et al. [37] presented adaptively-secure, non-interactive constructions based on the Decision Composite Residuosity (DCR) and the Learning-With-Errors (LWE) assumptions. More recently, [33, 53] presented very simple, practical, adaptively-secure, non-interactive constructions based on the DDH assump-

tion in the random oracle model.

Finally, the application of threshold decryption to encrypted mempools gives rise to several new concepts in threshold decryption, in addition to the new *decryption context* studied in this paper. One example is the need to trace corrupt validators who sell their decryption shares. This leads to the concept of *accountable threshold decryption* due to Boneh, Partap, and Rotem [15]. Another example is the need to threshold decrypt multiple transactions in a block. This leads to the concept of *batch threshold decryption*, due to Choudhuri, Garg, Piet, and Policharla [28], and developed further in [2, 29, 17, 55, 12]. We discuss this further in Section 9.

2 Preliminaries

2.1 Threshold decryption

We state here a definition for a very general notion of public-key threshold decryption scheme that includes the new notion of a **decryption context**.

Definition 1. *A **public-key threshold decryption scheme** $\mathsf{E} = (G, E, D, C)$ is a tuple of four efficient algorithms:*

- *G is a probabilistic **key generation algorithm** that is invoked as*

$$(pk, pkc, sk_1, \ldots, sk_N) \leftarrow\!\!\$\ G(N, t, f)$$

  *to generate N shares of a secret key with reconstruction threshold t and corruption bound f, where $N \geq t > f$. It outputs a **public key** pk, a **combiner public key** pkc, and N **decryption key shares** $sk_1, \ldots, sk_N$.*

- *E is a probabilistic **encryption algorithm** that is invoked as*

$$ctxt \leftarrow\!\!\$\ E(pk, m, ad),$$

  where pk is a public key output by G, m is a message, and ad is associated data.

- *D is a (possibly) probabilistic **decryption algorithm** that is invoked as*

$$ds_i \leftarrow\!\!\$\ D(sk_i, ctxt, ad, dc),$$

  where sk_i is one of the decryption key shares output by G, $ctxt$ is a ciphertext, ad is associated data, dc is a decryption context, and ds_i is a decryption share for $ctxt$ using sk_i.

- *C is a deterministic **combiner algorithm** that is invoked as*

$$m \leftarrow C(pkc, ctxt, ad, \boldsymbol{J}, \{ds_j\}_{j \in \boldsymbol{J}}, dc),$$

  where pkc is the combiner public key, $ctxt$ is a ciphertext, ad is associated data, $\boldsymbol{J}$ is a subset of $\{1, \ldots, N\}$ of size t, each ds_j is a decryption share of $ctxt$, and dc is a decryption context. The algorithm either outputs a plaintext m, the special symbol `reject`, or a special message $\texttt{blame}(\boldsymbol{J}^*)$, where $\boldsymbol{J}^*$ is a nonempty subset of $\boldsymbol{J}$.

  Intuitively, the message $\texttt{blame}(\boldsymbol{J}^*)$ indicates that the provided decryption shares ds_j for $j \in \boldsymbol{J}^*$ are invalid.

- ***Correctness:*** *as usual, decryption should correctly decrypt a properly constructed ciphertext; specifically, for all possible outputs* $(pk, pkc, sk_1, \ldots, sk_N)$ *of* $G(N, t, f)$*, all messages* m*, all associated data* ad*, all decryption contexts* dc*, all possible outputs* $ctxt$ *of* $E(pk, m, ad)$*, all* t*-size subsets* $\boldsymbol{J}$ *of* $\{1, \ldots, N\}$*, and all possible outputs* ds_j *of* $D(sk_j, ctxt, ad, dc)$ *for* $j \in \boldsymbol{J}$*, we have*

$$C\big(pkc,\ ctxt,\ ad,\ \boldsymbol{J},\ \{ds_j\}_{j\in\boldsymbol{J}},\ dc\big) = m.$$

In the above definition, messages lie in some finite message space $\boldsymbol{M}$, ciphertexts in some finite ciphertext space $\boldsymbol{Ctxt}$, associated data in some finite space $\boldsymbol{AD}$, and decryption contexts lie in some finite space $\boldsymbol{DC}$. We say that E is defined over $(\boldsymbol{M}, \boldsymbol{AD}, \boldsymbol{Ctxt}, \boldsymbol{DC})$. Also, just as for a secret sharing scheme, a threshold decryption scheme may impose constraints on the parameters N, t, and f (such as an upper bound on N and on the relationship between N, t, and f). For any particular scheme, we will refer to **allowable parameters** as those (N, t, f) tuples that are allowed by the scheme.

Remark 1 (Context-dependent and context-free schemes). *If the decryption context space* $\boldsymbol{DC}$ *contains more than one element, we say* E ***context dependent****. Otherwise, we say* E *is* ***context free****, and, by convention, we may naturally omit* dc *from the inputs to the algorithms* D *and* C*, and simply say that* E *is defined over* $(\boldsymbol{M}, \boldsymbol{AD}, \boldsymbol{Ctxt})$*.*

Remark 2 (Corruption parameter f). *The parameter* f *will be used in the security game as a bound on the number of secret keys that the adversary can request from the challenger at the beginning of the game. For our concrete construction in Section 3, we can set* $f = t - 1$*. For our generic construction in Section 6, we need to set* $f < t - 1$ *if the underlying legacy low-threshold CCA-secure decryption scheme has a lower reconstruction threshold than than the designed reconstruction threshold of the context-dependent high-threshold CCA-secure decryption scheme.*

2.1.1 Robustness.

Beyond security for the scheme, a threshold decryption scheme should also be robust against a malicious decryption server that is trying to disrupt the decryption process. We define two such robustness properties that any practical threshold decryption scheme should provide. Our approach to defining robustness closely follows that of [16].

Suppose a combiner receives a collection $\{ds_j\}_{j\in\boldsymbol{J}}$ of t decryption shares on a given ciphertext/associated data pair $(ctxt, ad)$. By Definition 1, the combiner will output either a plaintext $m \in \boldsymbol{M} \cup \{\texttt{reject}\}$, or a special message $\texttt{blame}(\boldsymbol{J}^*)$, where $\boldsymbol{J}^*$ is a nonempty subset of $\boldsymbol{J}$.

- If the output of the combiner is $\texttt{blame}(\boldsymbol{J}^*)$, then we would like it to be the case that all of the decryption shares ds_j for $j \in \boldsymbol{J}^*$ are "bad", in the sense that they were incorrectly generated by misbehaving decryption servers. A threshold signature scheme that guarantees that this (effectively) always happens is said to provide **accurate blaming**.

- Otherwise, if the output of the combiner is a plaintext m, we say that the collection of shares $\{ds_j\}_{j\in\boldsymbol{J}}$ is **valid** for $(ctxt, ad)$. We say that the threshold decryption scheme is **consistent** if it is infeasible to come up with any other valid collection of decryption shares that combines to yield a plaintext $m' \neq m$.

A threshold decryption scheme is **robust** if it satisfies both properties.

In practice, robustness allows a combiner who is trying to decrypt a ciphertext to proceed as follows. If the combiner algorithm outputs $\texttt{blame}(\boldsymbol{J}^*)$, then the combiner can discard the "bad" shares in $\boldsymbol{J}^*$, and seek out $t - |\boldsymbol{J}^*|$ "good" shares from among the remaining decryption servers. As long as there are t correctly behaving servers available, the accurate blaming property guarantees that the combiner can repeat this process until it gets a valid collection of shares, and the consistency property guarantees that when this happens, the resulting plaintext is the same that it would get from any other valid collection of decryption shares.

Also note that the consistency property, together with the correctness property in Definition 1, guarantees that if a ciphertext is a correctly generated encryption of a message, then any valid collection of decryption shares when combined will yield the original message. We now define these two properties formally.

The *accurate blaming* property is defined via the following attack game.

Attack Game 1 (accurate blaming). *For a given threshold decryption scheme* $\mathsf{E} = (G, E, D, C)$*, defined over* $(\boldsymbol{M}, \boldsymbol{AD}, \boldsymbol{Ctxt}, \boldsymbol{DC})$ *and a given adversary* $\mathfrak{A}$*, we define the following attack game.*

- *The adversary sends to the challenger polynomially-bounded allowable parameters* (N, t, f)*.*
- *The challenger runs* $(pk, pkc, sk_1, \ldots, sk_N) \leftarrow\!\!\$\; G(N, t, f)$ *and sends all this data to the adversary.*
- *The adversary sends to the challenger an index* $j^* \in \{1, \ldots, N\}$*, a ciphertext* $ctxt$*, associated data* ad*, and decryption context* dc*.*
- *The challenger runs* $ds \leftarrow\!\!\$\; D(sk_{j^*}, ctxt, ad, dc)$ *and sends* ds *to the adversary.*
- *The adversary outputs*
$$(\boldsymbol{J}, \{ds_j\}_{j \in \boldsymbol{J}}).$$
- *We say the adversary wins the game if*
  - $j^* \in \boldsymbol{J}$*,*
  - $ds_{j^*} = ds$*, and*
  - $C(pkc, ctxt, ad, \boldsymbol{J}, \{ds_j\}_{j \in \boldsymbol{J}}, dc) = \texttt{blame}(\boldsymbol{J}^*)$*, where* $j^* \in \boldsymbol{J}^*$*.*

We define $\mathfrak{A}$*'s advantage with respect to* E*, denoted* $\mathrm{blmPKEadv}[\mathfrak{A}, \mathsf{E}]$*, as the probability that* $\mathfrak{A}$ *wins the game.*

Definition 2 (accurate blaming). *We say that a threshold decryption scheme* E *provides* ***accurate blaming*** *if for all efficient adversaries* $\mathfrak{A}$*, the quantity* $\mathrm{blmPKEadv}[\mathfrak{A}, \mathsf{E}]$ *is negligible.*

Remark 3. *One could think of a definition in which the adversary can ask the challenger to partially decrypt many ciphertexts, and then break accurate blaming with respect to one of the challenger's responses. It is not hard to see that our definition is polynomially-equivalent to this more general definition via a standard query-guessing reduction.*

Remark 4. *One could also consider the notion of* perfect *accurate blaming, requiring that for any (even computationally unbounded) adversary* $\mathfrak{A}$*, its advantage* $\mathrm{blmPKEadv}[\mathfrak{A}, \mathsf{E}]$ *is* 0*. Looking ahead, our construction in Section 3 will satisfy this stronger notion. Our generic construction in Section 6 can also satisfy this perfect notion, depending on the underlying building blocks.*

The *consistency* property is defined via the following attack game.

Attack Game 2 (consistent threshold decryption). *For a given threshold decryption scheme* $\mathsf{E} = (G, E, D, C)$*, defined over* $(\boldsymbol{M}, \boldsymbol{AD}, \boldsymbol{Ctxt}, \boldsymbol{DC})$ *and a given adversary* $\mathfrak{A}$*, we define the following attack game.*

- *The adversary sends to the challenger polynomially-bounded allowable parameters* (N, t, f)*.*
- *The challenger runs* $(pk, pkc, sk_1, \ldots, sk_N) \leftarrow\!\!\$\; G(N, t, f)$ *and sends all this data to the adversary.*
- *The adversary outputs*

$$(ctxt, ad, \boldsymbol{J}_1, \{ds_{1j}\}_{j\in\boldsymbol{J}_1}, dc_1, \boldsymbol{J}_2, \{ds_{2j}\}_{j\in\boldsymbol{J}_2}, dc_2),$$

  where $ctxt \in \boldsymbol{Ctxt}$*,* $ad \in \boldsymbol{AD}$*,* $\boldsymbol{J}_1$ *and* $\boldsymbol{J}_2$ *are subsets of* $\{1, \ldots, N\}$ *of size* t*,* $\{ds_{1j}\}_{j\in\boldsymbol{J}_1}$ *and* $\{ds_{2j}\}_{j\in\boldsymbol{J}_2}$ *are collections of decryption shares, and* dc_1 *and* dc_2 *are decryption contexts.*

- *We say the adversary wins the game if*
  - $C(pkc, ctxt, ad, \boldsymbol{J}_1, \{ds_{1j}\}_{j\in\boldsymbol{J}_1}, dc_1) = m_1 \in \boldsymbol{M} \cup \{\texttt{reject}\}$*,*
  - $C(pkc, ctxt, ad, \boldsymbol{J}_2, \{ds_{2j}\}_{j\in\boldsymbol{J}_2}, dc_2) = m_2 \in \boldsymbol{M} \cup \{\texttt{reject}\}$*, and*
  - $m_1 \neq m_2$*.*

We define $\mathfrak{A}$*'s advantage with respect to* E*, denoted* $\mathrm{conPKEadv}[\mathfrak{A}, \mathsf{E}]$*, as the probability that* $\mathfrak{A}$ *wins the game.*

Definition 3 (consistent threshold decryption). *We say that a threshold decryption scheme* E *is* ***consistent*** *if for all efficient adversaries* $\mathfrak{A}$*, the quantity* $\mathrm{conPKEadv}[\mathfrak{A}, \mathsf{E}]$ *is negligible.*

One approach to achieve robustness (as given, for example, in [54]), is to introduce a **decryption share validation algorithm** V, which is invoked as $V(pkc, ctxt, j, ds_j, ad, dc)$ and returns either `accept` or `reject`. We say a decryption share ds_j is **valid for a given** $ctxt$ **(with respect to** j**,** ad**, and** dc**)** if $V(pkc, ctxt, j, ds_j, ad, dc) = \texttt{accept}$, and otherwise we say it is **invalid**.

The combiner algorithm on input

$$(pkc, ctxt, ad, \boldsymbol{J}, \{ds_j\}_{j\in\boldsymbol{J}}, dc)$$

first runs the decryption share validation algorithm on $(pkc, ctxt, j, ds_j, ad, dc)$ for each $j \in \boldsymbol{J}$, and if any of these outputs `reject`, the combiner outputs $\texttt{blame}(\boldsymbol{J}^*)$ for the set $\boldsymbol{J}^* \subseteq \boldsymbol{J}$ of offending indices. Otherwise, the combiner must output some message $m \in \boldsymbol{M} \cup \{\texttt{reject}\}$.

To satisfy the robustness requirement, it is necessary and sufficient that

- any output of the decryption algorithm is (effectively) always valid,
- it is infeasible to create two quorums of valid decryption shares that combine to different messages.

2.1.2 Security.

In a traditional threshold decryption scheme, as defined, for example, in [54], the security definition intuitively says following:

> if an adversary is able to obtain any information about an encrypted message, then *some* honest party must have generated a corresponding decryption share.

The above notion may be referred to as **low-threshold security**. Traditionally, a low-threshold scheme also insists that $t = f + 1$, but this is not strictly necessary. Indeed, our definition of low-threshold security more generally allows $t > f + 1$ but retains the same security property. This can be seen as the analog of "ramp secret sharing" for threshold decryption [6, 27]. Such a scheme only weakens the robustness properties of the scheme, since the reconstruction threshold t is greater than $f + 1$. Such a scheme provides no additional security, since an encrypted message is still guaranteed to remain private only if no honest party generates a corresponding decryption share.

The notion of **high-threshold security** actually ensures a stronger security property. Suppose that the set of corrupt parties is $\boldsymbol{L}$, where $|\boldsymbol{L}| \leq f$ and $\Delta \coloneqq t - |\boldsymbol{L}| > 0$. Intuitively, the security definition says that:

> if an adversary is able to obtain any information about an encrypted message, then *at least* Δ honest parties must have generated a corresponding decryption share *under the same decryption context.*

We first formally define high-threshold security, and then define low-threshold security as a restriction thereof. Our definitions here only model static corruptions (while adaptive corruptions are modeled in Section 8.1).

Attack Game 3 (high-threshold CCA security). *For a public-key threshold decryption scheme* $\mathsf{E} = (G, E, D, C)$ *defined over* $(\boldsymbol{M}, \boldsymbol{AD}, \boldsymbol{Ctxt}, \boldsymbol{DC})$*, and for a given adversary* $\mathfrak{A}$*, we define two experiments.*

Experiment b $(b = 0, 1)$**:**

- Setup: *the adversary sends polynomially-bounded allowable parameters* (N, t, f)*, and a subset* $\boldsymbol{L} \subseteq \{1, \ldots, N\}$*, where* $|\boldsymbol{L}| \leq f$ *and* $\Delta \coloneqq t - |\boldsymbol{L}| > 0$*, to the challenger.*

  The challenger does the following:

  - *initialize an empty associative array*

    $$Map : \boldsymbol{Ctxt} \times \boldsymbol{AD} \to 2^{\boldsymbol{DC} \times \{1,\ldots,N\} \setminus \boldsymbol{L}},$$

  - *run*

    $$(pk, pkc, sk_1, \ldots, sk_N) \leftarrow\!\!\$\ G(N, t, f),$$

  - *send* pk*,* pkc*, and* $\{sk_\ell\}_{\ell \in \boldsymbol{L}}$ *to* $\mathfrak{A}$*.*

- $\mathfrak{A}$ *then makes a series of queries to the challenger. Each query can be one of two types:*

– Encryption query: *each encryption query consists of a triple*

$$(m_0, m_1, ad) \in \boldsymbol{M}^2 \times \boldsymbol{AD},$$

where the messages m_0, m_1 are of the same length.
The challenger does the following:

* *compute* $ctxt \leftarrow\!\!\$\ E(pk, m_b, ad)$,
* *if* $(ctxt, ad) \notin \mathsf{Domain}(Map)$, *then initialize* $Map[ctxt, ad] \leftarrow \emptyset$,
* *send* $ctxt$ *to* $\mathfrak{A}$.

– Decryption query: *each decryption query consists of a tuple*

$$(ctxt, ad, dc, i) \in \boldsymbol{Ctxt} \times \boldsymbol{AD} \times \boldsymbol{DC} \times \{1, \ldots, N\} \setminus \boldsymbol{L}$$

such that either
(D1) $(ctxt, ad) \notin \mathsf{Domain}(Map)$, *or*
(D2) $(ctxt, ad) \in \mathsf{Domain}(Map)$, $(dc, i) \notin Map[ctxt, ad]$, *and*

$$|\{j : (dc, j) \in Map[ctxt, ad]\}| < \Delta - 1$$

The challenger does the following:

* *if* $(ctxt, ad) \in \mathsf{Domain}(Map)$, *then update*

$$Map[ctxt, ad] \leftarrow Map[ctxt, ad] \cup \{(dc, i)\},$$

* *compute* $D(sk_i, ctxt, ad, dc)$ *and send this to* $\mathfrak{A}$.

- *At the end of the game,* $\mathfrak{A}$ *outputs a bit* $\overline{b} \in \{0, 1\}$.

If W_b is the event that $\mathfrak{A}$ *outputs 1 in Experiment b, define* $\mathfrak{A}$*'s* ***advantage*** *with respect to* E *as*

$$\mathrm{hthCCAadv}[\mathfrak{A}, \mathsf{E}] \coloneqq \Big|\Pr[W_0] - \Pr[W_1]\Big|.$$

Definition 4 (high-threshold CCA security). *A public-key threshold decryption scheme* E *is* ***high-threshold CCA secure*** *if for all efficient adversaries* $\mathfrak{A}$*, the value* $\mathrm{hthCCAadv}[\mathfrak{A}, \mathsf{E}]$ *is negligible.*

To define **low-threshold security**, we first define a modified version of Attack Game 3 in which we simply drop the disjunct (D2) in the precondition of the decryption queries. This restricts the actions of the adversary and therefore characterizes a weaker form of security. We denote the adversary's advantage in this game $\mathrm{lthCCAadv}[\mathfrak{A}, \mathsf{E}]$.

Definition 5 (low-threshold CCA security). *A public-key threshold decryption scheme* E *is* ***low-threshold CCA secure*** *if for all efficient adversaries* $\mathfrak{A}$*, the value* $\mathrm{lthCCAadv}[\mathfrak{A}, \mathsf{E}]$ *is negligible.*

Remark 5. *In defining low-threshold security, the decryption context plays no role. This can be seen formally by observing that once we eliminate the disjunct (D2) from the decryption precondition, decryption contexts do not have any material impact on the attack game. Informally, the reason is that an encrypted message is no longer guaranteed to remain private as soon as one honest party generates a corresponding decryption share with respect to* any *decryption context — whether or not additional honest parties generate such shares with the same or different contexts is irrelevant. Therefore, we may as well assume that such a threshold decryption scheme is context free.*

Remark 6. *The above definition is more general than the traditional definition [54], in that it allows $t > f + 1$, which (as remarked above) would be the analog of "ramp secret sharing". For a traditional low-threshold scheme, we would have $t = f + 1$.*

Remark 7. *The above definitions of CCA security are defined in terms of attack games, each with two experiments. As is standard (see Section 2.2.5 of [16]), we can also define a "bit guessing" versions of these attack games, in which the challenger chooses $b \in \{0, 1\}$ at random, and runs Experiment b. The corresponding advantage is defined to be the distance between $1/2$ and the probability that $\overline{b} = b$, and is denoted* $\mathrm{hthCCAadv}^*[\mathfrak{A}, \mathsf{E}]$ *(respectively,* $\mathrm{lthCCAadv}^*[\mathfrak{A}, \mathsf{E}]$*), and the two-experiment advantage is always equal to twice the bit-guessing advantage.*

3 A construction

We present here a construction that is a variant of the TDH1 scheme in [54], which we call $\mathsf{E}_{\mathrm{htdh1}}$. The encryption algorithm of $\mathsf{E}_{\mathrm{htdh1}}$, is essentially identical to that of TDH1 — the only difference is that we use a more general symmetric encryption algorithm. The decryption and combiner algorithms are just slightly tweaked to incorporate decryption contexts.

While our general definition of a threshold decryption scheme requires the specification of both a reconstruction threshold t and a corruption bound f, the scheme $\mathsf{E}_{\mathrm{htdh1}}$ actually makes no use of the parameter f, and we can in fact assume that $f = t - 1$.

Note that THD1 was proved CCA secure in [54] in the random oracle model under the computational Diffie-Hellman (CDH) assumption. The paper [54] did not consider high-threshold CCA security or decryption contexts (although it did consider associated data). It does not seem possible to prove the security of $\mathsf{E}_{\mathrm{htdh1}}$ in the high-threshold setting under the same assumption as [54]. However, we can prove it is secure (again, in the ROM) under the **linear one-more Diffie-Hellman (LOMDH)** assumption, which is a falsifiable assumption that can be justified in the generic group model (GGM), and which is the same assumption used to analyze high-threshold BLS signatures. See Section 4 for a formal definition and GGM analysis of the LOMDH assumption.

The scheme $\mathsf{E}_{\mathrm{htdh1}} = (G, E, D, C)$ is parameterized in terms of a message space $\boldsymbol{M}$, associated data space $\boldsymbol{AD}$, and decryption context space $\boldsymbol{DC}$, and makes use of the following:

- a group $\mathbb{G}$ of prime order q generated by $\mathcal{G} \in \mathbb{G}$; we write $\mathbb{G}$ additively, with $\mathcal{O} \in \mathbb{G}$ denoting the identity element;
- a symmetric cipher E_{s} (which will be assumed semantically secure) with deterministic encryption algorithm E_{s}, deterministic decryption algorithm D_{s}, message space $\boldsymbol{M}$, key space $\boldsymbol{K}$, and ciphertext space $\boldsymbol{C}$; we write $c \leftarrow E_{\mathrm{s}}(k, m)$ to encrypt m and obtain the ciphertext c, and $m \leftarrow D_{\mathrm{s}}(k, c)$ to decrypt c and obtain m;

- various hash functions (which will be modeled as random oracles):
  - for *key derivation*, $H_{\mathrm{kd}} : \mathbb{G} \times \mathbb{G} \to \boldsymbol{K}$,
  - for *encryption group derivation*, $H_{\mathrm{egd}} : \mathbb{G} \times \mathbb{G} \times \boldsymbol{AD} \times \boldsymbol{C} \to \mathbb{G}$,
  - for *encryption challenge derivation*, $H_{\mathrm{ecd}} : \mathbb{G}^3 \to \mathbb{Z}_q$,
  - for *decryption group derivation*, $H_{\mathrm{dgd}} : \boldsymbol{AD} \times \boldsymbol{DC} \times (\mathbb{G} \times \mathbb{G} \times \mathbb{Z}_q \times \mathbb{Z}_q \times \boldsymbol{C}) \to \mathbb{G}$,
  - for *decryption challenge derivation*, $H_{\mathrm{dcd}} : \mathbb{G}^7 \to \mathbb{Z}_q$.

In what follows, it will be convenient to define the **Diffie-Hellman operator** $\mathbb{G} \times \mathbb{G} \to \mathbb{G}$, defined by $\mathsf{DH}(x\mathbb{G}, y\mathbb{G}) \coloneqq xy\mathbb{G}$.

Key generation. The key generation algorithm G works as follows. It generates $x \in \mathbb{Z}_q$ at random, and generates a random t-out-of-N Shamir sharing $(x_1, \ldots, x_N)$ of x. It also generates a random t-out-of-N Shamir sharing $(z_1, \ldots, z_N)$ of 0. It then computes

$$\mathcal{X} \leftarrow x\mathcal{G}$$

and for $i = 1, \ldots, N$ it computes

$$\mathcal{X}_i \leftarrow x_i\mathcal{G} \quad \text{and} \quad \mathcal{Z}_i \leftarrow z_i\mathcal{G}.$$

Finally, it outputs $(pk, pkc, sk_1, \ldots, sk_N)$, where

- $pk \coloneqq \mathcal{X}$,
- $pkc \coloneqq (\mathcal{X}; \mathcal{X}_1, \ldots, \mathcal{X}_N; \mathcal{Z}_1, \ldots, \mathcal{Z}_N)$, and
- $sk_i \coloneqq (x_i, z_i)$ for $i = 1, \ldots, N$.

Encryption. The encryption algorithm E takes as input a public key $pk = \mathcal{X} \in \mathbb{G}$, a message $m \in \boldsymbol{M}$, and associated data $ad \in \boldsymbol{AD}$, and runs as follows.

$r \leftarrow\!\!\$\, \mathbb{Z}_q$, $\mathcal{R} \leftarrow r\mathcal{G}$, $\mathcal{U} \leftarrow r\mathcal{X}$, $k \leftarrow H_{\mathrm{kd}}(\mathcal{R}, \mathcal{U}) \in \boldsymbol{K}$, $c \leftarrow E_{\mathrm{s}}(k, m) \in \boldsymbol{C}$
$r' \leftarrow\!\!\$\, \mathbb{Z}_q$, $\mathcal{R}' \leftarrow r'\mathcal{G}$
$\mathcal{Y} \leftarrow H_{\mathrm{egd}}(\mathcal{R}, \mathcal{R}', ad, c)$, $\mathcal{V} \leftarrow r\mathcal{Y}$, $\mathcal{V}' \leftarrow r'\mathcal{Y}$
$e \leftarrow H_{\mathrm{ecd}}(\mathcal{Y}, \mathcal{V}, \mathcal{V}') \in \mathbb{Z}_q$, $r'' \leftarrow r' + re \in \mathbb{Z}_q$
output $ctxt \coloneqq (\mathcal{R}, \mathcal{V}, e, r'', c)$

Intuition. The first line of the encryption algorithm computes a hybrid ElGamal ciphertext $(\mathcal{R}, c)$, where $\mathcal{X}$ is the public key, $\mathcal{R}$ is the encryptor's ephemeral public key $\mathcal{R}$, and the symmetric ciphertext c is computed by deriving a symmetric encryption key k from $\mathcal{U} = \mathsf{DH}(\mathcal{R}, \mathcal{X})$ — including the group element $\mathcal{R}$ in the hash H_{kd} is not strictly necessary, but allows for a tighter security reduction. The remaining lines compute a zero-knowledge proof (e, r'') that $\mathcal{V} = \mathsf{DH}(\mathcal{R}, \mathcal{Y})$. Here, $\mathcal{Y}$ is a group element derived from the hash function H_{egd}, and the challenge e for the underlying Sigma protocol is derived from the hash function H_{ecd}. The exact way in which the inputs to these hash functions are defined follows the same logic as in the TDH1 scheme in [54] to ensure non-malleability.

Decryption. The decryption algorithm D as run by party P_i takes as input a secret key $sk_i = (x_i, z_i) \in \mathbb{Z}_q \times \mathbb{Z}_q$, a ciphertext $ctxt = (\mathcal{R}, \mathcal{V}, e, r'', c) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_q \times \mathbb{Z}_q \times \boldsymbol{C}$, associated data $ad \in \boldsymbol{AD}$, and decryption context $dc \in \boldsymbol{DC}$, and runs as follows. First, it checks that

$$\begin{gathered} e = H_{\mathrm{ecd}}(\mathcal{Y}, \mathcal{V}, \mathcal{V}'), \quad \text{where} \\ \mathcal{R}' = r''\mathcal{G} - e\mathcal{R}, \quad \mathcal{Y} = H_{\mathrm{egd}}(\mathcal{R}, \mathcal{R}', ad, c), \quad \mathcal{V}' = r''\mathcal{Y} - e\mathcal{V}. \end{gathered} \tag{1}$$

If this check fails, it outputs `reject`. Otherwise, it proceeds as follows:

$\mathcal{X}_i \leftarrow x_i\mathcal{G}, \;\; \mathcal{Z}_i \leftarrow z_i\mathcal{G}$ *// these may also be stored as part of the secret key sk_i*
$\mathcal{S} \leftarrow H_{\mathrm{dgd}}(ad, dc, ctxt) \in \mathbb{G}$
$\mathcal{W}_i \leftarrow x_i\mathcal{R} + z_i\mathcal{S}$
$x'_i, z'_i \leftarrow\!\!\$\, \mathbb{Z}_q, \quad \mathcal{X}'_i \leftarrow x'_i\mathcal{G}, \quad \mathcal{Z}'_i \leftarrow z'_i\mathcal{G}, \quad \mathcal{W}'_i \leftarrow x'_i\mathcal{R} + z'_i\mathcal{S}$
$e_i \leftarrow H_{\mathrm{dcd}}(\mathcal{S}, \mathcal{X}_i, \mathcal{Z}_i, \mathcal{W}_i, \mathcal{X}'_i, \mathcal{Z}'_i, \mathcal{W}'_i)$
$x''_i \leftarrow x'_i + e_i x_i \in \mathbb{Z}_q, \quad z''_i \leftarrow z'_i + e_i z_i \in \mathbb{Z}_q$
output $ds_i := (\mathcal{W}_i, e_i, x''_i, z''_i)$

Intuition. The check (1) verifies the proof that $\mathcal{V} = \mathsf{DH}(\mathcal{R}, \mathcal{Y})$. In what follows, the group element $\mathcal{S}$ is derived via a hash function (to be modeled as a random oracle) in a way that depends not only on c and ad, but also on the decryption context dc. Then, the group element $\mathcal{W}_i = x_i\mathcal{R} + z_i\mathcal{S}$ is formed, and what follows is the construction of a standard zero-knowledge proof (e_i, x''_i, z''_i) that $\mathcal{W}_i$ was formed correctly.

Decryption share validation. A decryption share ds_i from P_i is validated with respect to $pkc = (\mathcal{X}; \mathcal{X}_1, \ldots, \mathcal{X}_N; \mathcal{Z}_1, \ldots, \mathcal{Z}_N)$, $ctxt = (\mathcal{R}, e, r'', c)$, ad, and dc by first checking if (1) holds. If this does not hold, then ds_i is valid if and only if it is equal to `reject`. Otherwise, ds_i is valid if and only if $(\mathcal{W}_i, e_i, x''_i, z''_i)$ satisfies

$$\begin{gathered} e_i = H_{\mathrm{dcd}}(\mathcal{S}, \mathcal{X}_i, \mathcal{Z}_i, \mathcal{W}_i, \mathcal{X}'_i, \mathcal{Z}'_i, \mathcal{W}'_i), \quad \text{where} \\ \mathcal{S} = H_{\mathrm{dgd}}(ad, dc, ctxt), \\ \mathcal{X}'_i = x''_i\mathcal{G} - e_i\mathcal{X}_i, \quad \mathcal{Z}'_i = z''_i\mathcal{G} - e_i\mathcal{Z}_i, \quad \mathcal{W}'_i = x''_i\mathcal{R} + z''_i\mathcal{S} - e_i\mathcal{W}_i. \end{gathered} \tag{2}$$

Intuition. The check (2) is essentially just a verification of the zero-knowledge proof that $\mathcal{W}_i$ was formed correctly.

Combining decryption shares. Suppose we are given a quorum $\{ds_j\}_{j \in \boldsymbol{J}}$ of decryption shares, together with $pkc = (\mathcal{X}; \mathcal{X}_1, \ldots, \mathcal{X}_N; \mathcal{Z}_1, \ldots, \mathcal{Z}_N)$, $ctxt = (\mathcal{R}, e, r'', c)$, ad, and dc. We first run the decryption share validation algorithm above to determine if all decryption shares are valid. If not, we output $\mathtt{blame}(\boldsymbol{J}^*)$, where $\boldsymbol{J}^*$ is the subset of $\boldsymbol{J}$ for which ds_j is invalid for all $j \in \boldsymbol{J}^*$. Otherwise, if all decryption shares are equal to `reject`, we also output `reject`. Otherwise, each decryption share ds_j is of the form $(\mathcal{W}_j, \ldots)$, and we compute

$$\mathcal{U} \leftarrow \sum_{j \in \boldsymbol{J}} \lambda_j^{(\boldsymbol{J})} \mathcal{W}_j,$$

where $\{\lambda_j^{(\boldsymbol{J})}\}_{j \in \boldsymbol{J}}$ is the collection of interpolation coefficients that recovers the secret from a t-out-of-N Shamir sharing from the shares specified by the index set $\boldsymbol{J}$. Finally, we compute $k \leftarrow H_{\mathrm{kd}}(\mathcal{R}, \mathcal{U}) \in \boldsymbol{K}$ and output $m \leftarrow D_{\mathrm{s}}(k, c) \in \boldsymbol{M}$.

Intuition. Since the share validation algorithm ensures that each $\mathcal{W}_j$ is of the form $x_j\mathcal{R} + z_j\mathcal{S}$, we have

$$\mathcal{U} = \sum_{j\in\boldsymbol{J}} \lambda_j^{(\boldsymbol{J})}\mathcal{W}_j = \sum_{j\in\boldsymbol{J}} \lambda_j^{(\boldsymbol{J})}(x_j\mathcal{R} + z_j\mathcal{S}) = \sum_{j\in\boldsymbol{J}} \lambda_j^{(\boldsymbol{J})}x_j\mathcal{R} + \sum_{j\in\boldsymbol{J}} \lambda_j^{(\boldsymbol{J})}z_j\mathcal{S} = x\mathcal{R}.$$

4 The linear one-more Diffie-Hellman assumption

We give here a formal definition of the **linear one-more Diffie-Hellman (LOMDH) assumption**. This is a trivial generalization of an assumption introduced in [42] (and later [5]) to prove the security of high-threshold BLS signatures [14]. The assumption states that it is infeasible for an adversary to win the following attack game. In this game, a challenger first chooses $s_1, \ldots, s_k$ and $t_1, \ldots, t_\ell$ at random from $\mathbb{Z}_q$, and gives to the adversary the group elements

$$\mathcal{S}_i \coloneqq s_i\mathcal{G} \in \mathbb{G} \quad (i = 1, \ldots, k)$$

and

$$\mathcal{T}_j \coloneqq t_j\mathcal{G} \in \mathbb{G} \quad (j = 1, \ldots, \ell).$$

Next, the adversary makes a sequence of **linear DH-queries**. Each such query consists of a matrix of scalars $\{\kappa_{i,j}\} \in \mathbb{Z}_q^{[k]\times[\ell]}$, to which the challenger responds with the group element

$$\sum_{i,j} \kappa_{i,j}s_it_j\mathcal{G} = \sum_{i,j} \kappa_{i,j}\mathsf{DH}(\mathcal{S}_i, \mathcal{T}_j).$$

At the end of the game, the adversary outputs a list of pairs, each of the form

$$(\mathcal{V}, \{\kappa_{i,j}\}) \in \mathbb{G} \times \mathbb{Z}_q^{[k]\times[\ell]}.$$

The adversary wins the game if for one such output group/matrix pair, we have

(i) $\{\kappa_{i,j}\}$ is not in the linear span of the input matrices, and

(ii)
$$\mathcal{V} = \sum_{i,j} \kappa_{i,j}s_it_j\mathcal{G} = \sum_{i,j} \kappa_{i,j}\mathsf{DH}(\mathcal{S}_i, \mathcal{T}_j)$$

We shall refer to the group elements $\mathcal{S}_1, \ldots, \mathcal{S}_k$ in the attack game as **row elements** and $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$ as **column elements**. Each pair in the adversary's output list asserts a **relation** as in condition (ii) above. The adversary only wins the game if such an assertion holds that is not **trivial**, in the sense of condition (i) above.

Note that the attack games defined in [42, 5] are the same as the above game with the restriction that $k = 1$ and the output list has length 1. The paper [5] shows that in this restricted setting, in the generic group model (GGM), the adversary's wins this game with probability $O(M^2/q)$, where M is a bound on the total number of queries. It is entirely straightforward to generalize this analysis to the more general setting where we lift the restrictions on k and the length of the output list, and M is a bound on the number of queries plus the length of the output list.

We believe our formulation of the LOMDH assumption is quite natural and has several advantages. First, it allows one to prove the security of high-threshold BLS signatures with a very tight

security reduction. Second, it has more applications — not the least of which is our analysis of a high-threshold decryption scheme.

It turns out that the LOMDH assumption is implied by the decisional Diffie-Hellman (DDH) assumption:

Theorem 1. *The DDH assumption implies the LOMDH assumption.*

Proof (sketch). This follows almost immediately from Exercise 10.12 in [16], which says that under the DDH the distribution $(\{\mathcal{S}_i\}_i, \{\mathcal{T}_j\}_j, \{\mathsf{DH}(\mathcal{S}_i, \mathcal{T}_j)\}_{i,j})$ is computationally indistinguishable from $(\{\mathcal{S}_i\}_i, \{\mathcal{T}_j\}_j, \{\mathcal{R}_{i,j}\}_{i,j})$ for random $\mathcal{R}_{i,j}$. □

This theorem, combined with Theorem 2 below, implies that our scheme $\mathsf{E}_{\text{htdh1}}$ is high-threshold CCA secure under the DDH (in the random oracle model). However, LOMDH is a computational assumption, and so $\mathsf{E}_{\text{htdh1}}$ may also be secure in groups where the DDH assumption is false.

5 Security analysis of $\mathsf{E}_{\text{htdh1}}$

Theorem 2. *If all hash functions are modeled as random oracles, if* E_{s} *is semantically secure, and if the LOMDH assumption holds, then* $\mathsf{E}_{\text{htdh1}}$ *is high-threshold CCA secure.*

Proof. We make an argument based on a sequence of games, starting from the bit-guessing version of Attack Game 3 (see Remark 7), adapted to model all hash functions as random oracles, so that the adversary also queries the challenger to obtain the value of the random oracles at inputs of its choosing. Call this Game 0. Let $\boldsymbol{L}$ be the set of corrupt parties as chosen by the adversary and let $\Delta := t - |\boldsymbol{L}| > 0$.

In each Game j, we denote by p_j the probability that the adversary guesses the hidden bit b of the challenger.

Game 1. We now make a purely conceptual modification to Game 0, changing the way the key generation works. After the adversary specifies the set $\boldsymbol{L}$, we choose a set $\boldsymbol{L}' \supseteq \boldsymbol{L}$ of size exactly $t-1$. We generate x and x_j for $j \in \boldsymbol{L}'$, and then compute the remaining x_i's by interpolation:

$$x_i \leftarrow \lambda^{(i)} x + \sum_{j \in \boldsymbol{L}'} \lambda_j^{(i)} x_j,$$

for appropriate interpolation coefficients $\lambda^{(i)}$ and $\{\lambda_j^{(i)}\}_{j \in \boldsymbol{L}'}$. Similarly, we generate z_j at random for $j \in \boldsymbol{L}'$ and then compute the remaining z_i's by interpolation:

$$z_i \leftarrow \sum_{j \in \boldsymbol{L}'} \lambda_j^{(i)} z_j.$$

Let us call this Game 1. We have $p_1 = p_0$.

Looking ahead, in the reduction to LOMDH, the LOMDH adversary will run with full knowledge of $\{x_j\}_{j \in \boldsymbol{L}'}$ and $\{z_j\}_{j \in \boldsymbol{L}}$, while the group elements $\mathcal{X}$ and $\{\mathcal{Z}_j\}_{j \in \boldsymbol{L}' \setminus \boldsymbol{L}}$ will play the role of column elements in the LOMDH attack, and various group elements $\mathcal{R}$ and $\mathcal{S}$ corresponding to ciphertexts produced by the encryption oracle will play the role of row elements. To this end, we want to be able to simulate decryption queries that are not related to any encryption queries, knowing

only $\{x_j\}_{j\in\boldsymbol{L}'}$ and $\{z_j\}_{j\in\boldsymbol{L}}$, as well as the group elements $\mathcal{X}$ and $\{\mathcal{Z}_j\}_{j\in\boldsymbol{L}'\setminus\boldsymbol{L}}$. The next few games prepare the groundwork for this simulation.

Now consider a ciphertext $ctxt = (\mathcal{R}, \mathcal{V}, e, r'', c)$ generated by the encryption oracle with associated data ad and a ciphertext $\overline{ctxt} = (\overline{\mathcal{R}}, \overline{\mathcal{V}}, \overline{e}, \overline{r}'', \overline{c})$ submitted to the decryption oracle with associated data $\overline{ad}$. Clearly, the check (1) holds for $(ctxt, ad)$, so let $\mathcal{R}' = r''\mathcal{G} - e\mathcal{R}$ as in (1). If this check does not hold for $(\overline{ctxt}, \overline{ad})$, the decryption will result in `reject`. So suppose this check does hold, and let $\overline{\mathcal{R}}' = \overline{r}''\mathcal{G} - \overline{e}\overline{\mathcal{R}}$ as in (1). Then with overwhelming probability, we must have

$$(\mathcal{R}, \mathcal{R}', ad, c) = (\overline{\mathcal{R}}, \overline{\mathcal{R}}', \overline{ad}, \overline{c}) \quad \Longrightarrow \quad (ctxt, ad) = (\overline{ctxt}, \overline{ad}). \tag{3}$$

Note that the tuples $(\mathcal{R}, \mathcal{R}', ad, c)$ and $(\overline{\mathcal{R}}, \overline{\mathcal{R}}', \overline{ad}, \overline{c})$ are the inputs to H_{egd}. This fact is proven in [54] for the TDH1 scheme, and the proof carries over without change here.

Game 2. Based on the above, we modify Game 1 so that if for a given decryption query $(\overline{ctxt}, \overline{ad})$ we have (1) but not (3) for some $(ctxt, ad) \in \mathsf{Domain}(Map)$, the challenger returns `reject` without processing the ciphertext any further. We call this Game 2. The quantity $|p_2 - p_1|$ is negligible.

Game 3. We next modify Game 2 to program H_{egd} and H_{dgd} in a certain way. Without loss of generality, we may assume that whenever the adversary makes a decryption query, the random oracle queries to H_{egd} and H_{dgd} needed to process the decryption query have already been made directly by the adversary — if not, we simply modify the adversary to do so, as all of the inputs to these random oracle queries can be computed based on public data. Now suppose the adversary directly queries H_{egd} at an input $(\mathcal{R}, \mathcal{R}', ad, c)$ that has not previously been queried, either directly by the adversary or indirectly through an encryption query. Then the challenger programs H_{egd} by generating $y \in \mathbb{Z}_q$ at random and setting $H_{\mathrm{egd}}(\mathcal{R}, \mathcal{R}', ad, c) \coloneqq \mathcal{X} + y\mathcal{G}$. Furthermore, if any future encryption query evaluates H_{egd} at $(\mathcal{R}, \mathcal{R}', ad, c)$, the attack game aborts (and the adversary outputs some arbitrary value) — this happens with negligible probability. Additionally, suppose the adversary directly queries H_{dgd} at an input $(ad, dc, ctxt)$ such that $(ctxt, ad) \notin \mathsf{Domain}(Map)$ at the time of the query. Then the challenger programs H_{dgd} by generating $s \in \mathbb{Z}_q$ at random and setting $H_{\mathrm{dgd}}(ad, dc, ctxt) \coloneqq s\mathcal{G}$. Furthermore, if $(ctxt, ad)$ is ever added to $\mathsf{Domain}(Map)$ by an encryption query, the attack game aborts — again, this happens with negligible probability. We call this Game 3. The quantity $|p_3 - p_2|$ is negligible.

Game 4. We next modify Game 3 so that we replace the generation of the zero-knowledge proof (e_i'', x_i'', z_i'') in each decryption is replaced by the usual zero-knowledge simulator. This simulation involves programming the random oracle H_{dcd} and may fail with negligible probability. We call this Game 4. The quantity $|p_4 - p_3|$ is negligible.

Game 5. We next modify Game 4 so that (as promised) we can simulate decryption queries that are not related to any encryption queries, knowing only $\{x_j\}_{j\in\boldsymbol{L}'}$ and $\{z_j\}_{j\in\boldsymbol{L}}$, as well as the group elements $\mathcal{X}$ and $\{\mathcal{Z}\}_{j\in\boldsymbol{L}'\setminus\boldsymbol{L}}$. Suppose the adversary makes a decryption query $(ctxt, ad, dc, i)$ and $(ctxt, ad) \notin \mathsf{Domain}(Map)$. Let $ctxt = (\mathcal{R}, \mathcal{V}, e, r'', c)$ and assume that the check (1) holds with $\mathcal{R}'$, $\mathcal{Y}$, $\mathcal{V}'$ also as in (1). Also let $\mathcal{S} = H_{\mathrm{dgd}}(ad, dc, ctxt)$. By the rule imposed in Game 3 and random oracle programming done in Game 4, we have $\mathcal{Y} = \mathcal{X} + y\mathcal{G}$ where y is known to us. Also by the random oracle programming done in Game 4, we have $\mathcal{S} = s\mathcal{G}$ where s is known to us. By the soundness of the proof that $\mathcal{V} = \mathsf{DH}(\mathcal{R}, \mathcal{Y})$, with overwhelming probability, we have and therefore

$$\mathcal{V} = \mathsf{DH}(\mathcal{R}, \mathcal{Y}) = \mathsf{DH}(\mathcal{R}, \mathcal{X} + y\mathcal{G}) = \mathsf{DH}(\mathcal{R}, \mathcal{X}) + y\mathcal{R},$$

which means we can compute $\mathsf{DH}(\mathcal{R}, \mathcal{X})$ as $\mathcal{V} - y\mathcal{R}$.

So to carry out the simulated decryption, we need to compute

$$\begin{aligned}
\mathcal{W}_i &= \mathsf{DH}(\mathcal{R}, \mathcal{X}_i) + \mathsf{DH}(\mathcal{S}, \mathcal{Z}_i) \\
&= \mathsf{DH}\Big(\mathcal{R}, \lambda^{(i)}\mathcal{X} + \sum_{j \in \boldsymbol{L}'} \lambda_j^{(i)} x_j \mathcal{G}\Big) + \mathsf{DH}\Big(\mathcal{S}, \sum_{j \in \boldsymbol{L}'} \lambda_j^{(i)} \mathcal{Z}_j\Big) \\
&= \Big(\lambda^{(i)}\mathsf{DH}(\mathcal{R}, \mathcal{X}) + \sum_{j \in \boldsymbol{L}'} \lambda_j^{(i)} x_j \mathcal{R} \Big) + \Big(\sum_{j \in \boldsymbol{L}} \lambda_j^{(i)} z_j \mathcal{S} + \sum_{j \in \boldsymbol{L}' \setminus \boldsymbol{L}} \lambda_j^{(i)} \mathsf{DH}(\mathcal{S}, \mathcal{Z}_j) \Big) \\
&= \lambda^{(i)}(\mathcal{V} - y\mathcal{R}) + \sum_{j \in \boldsymbol{L}'} \lambda_j^{(i)} x_j \mathcal{R} + \sum_{j \in \boldsymbol{L}} \lambda_j^{(i)} s z_j \mathcal{G} + \sum_{j \in \boldsymbol{L}' \setminus \boldsymbol{L}} \lambda_j^{(i)} s \mathcal{Z}_j,
\end{aligned}$$

which we can do given the data provided. The rest of the output produced by the decryption oracle comes from the zero-knowledge simulator introduced in Game 4. We call this Game 5. The quantity $|p_5 - p_4|$ is negligible.

Game 6. We next modify Game 5 so that in the encryption queries, we replace the zero-knowledge proofs (e, r'') with simulations. Since the proofs here are slightly nonstandard, we specify exactly how this is done.

1. We choose $e, r'' \in \mathbb{Z}_q$ at random.
2. We compute $\mathcal{R}' \leftarrow r''\mathcal{G} - e\mathcal{R}$.
3. We check if H_{egd} has been evaluated at $(\mathcal{R}, \mathcal{R}', ad, c)$ — if so, we abort and if not, we program H_{egd} so that $H_{\text{egd}}(\mathcal{R}, \mathcal{R}', ad, c) \coloneqq \mathcal{Y} \coloneqq y\mathcal{G}$ for random $y \in \mathcal{R}$.
4. We set $\mathcal{V} \coloneqq y\mathcal{R} = \mathsf{DH}(\mathcal{R}, \mathcal{Y})$ as required (so the statement we are proving is, in fact, true) and set $\mathcal{V}' \coloneqq r''\mathcal{G} - e\mathcal{V}$.
5. We then check if H_{ecd} has been evaluated at $(\mathcal{Y}, \mathcal{V}, \mathcal{V}')$ — if so, we abort and if not, we program H_{ecd} so that $H_{\text{egd}}(\mathcal{Y}, \mathcal{V}, \mathcal{V}') \coloneqq e$.

These simulations may fail, but only with negligible probability. We call this Game 6. The quantity $|p_6 - p_5|$ is negligible.

Note that in Game 6, no two encryption queries will attempt to add the same pair $(ctxt, ad)$ to $\mathsf{Domain}(Map)$.

Game 7. We next modify Game 6 so that in each encryption query, we replace the symmetric key k, which is normally computed as $H_{\text{kd}}(\mathcal{R}, \mathcal{U})$, by a random element of the symmetric key space $\boldsymbol{K}$. Let us call the Game 7. The quantity $|p_7 - p_6|$ is bounded by the probability of the following failure event:

> the adversary evaluates the random oracle H_{kd} at one of these inputs $(\mathcal{R}, \mathcal{U})$ in Game 7.

We shall show below that this probability is bounded by the advantage of a certain adversary in breaking the LOMDH assumption, which implies that this probability is negligible. Moreover, under the semantic security assumption for E_{s}, it follows that $|p_7 - 1/2|$ is negligible, and we conclude that $|p_0 - 1/2|$ is negligible as well.

The LOMDH adversary. To complete the proof, we describe the above-mentioned LOMDH adversary. The LOMDH runs Game 7 above, with the following modifications. The LOMDH adversary will generate at random $x_j \in \mathbb{Z}_q$ at random for $j \in \boldsymbol{L}'$ and $z_j \in \mathbb{Z}_q$ for $j \in \boldsymbol{L}$. The group elements $\mathcal{X}$ and $\{\mathcal{Z}_j\}_{j \in \boldsymbol{L}' \setminus \boldsymbol{L}}$ are column elements obtained from the challenger in the LOMDH attack game. The row elements in the LOMDH attack game will be mapped to

- the group elements $\mathcal{R}$ that would normally be generated at random in each encryption query in Game 7, and
- the outputs of $\mathcal{S}$ that would normally be generated as the output of random oracle queries of the form $H_{\mathrm{dgd}}(ad, dc, ctxt)$ where $(ctxt, ad) \in \mathsf{Domain}(Map)$ at the time of the random oracle query.

Note that by the rules imposed in Game 3, these H_{dgd} queries will be made explicitly by the CCA adversary prior to any decryption query. Also note that by the rules imposed in Game 6, each such query $H_{\mathrm{dgd}}(ad, dc, ctxt)$ will correspond to the unique encryption query that added $(ctxt, ad)$ to $\mathsf{Domain}(Map)$. Also note that by the rules imposed in Game 6, the encryption queries can be processed with the given information, that is, knowing just the corresponding group element $\mathcal{R}$ obtained from the LOMDH.

Now consider a decryption query $(ctxt, ad, dc, i)$. In Game 4, we already introduced rules to process such a query when $(ctxt, ad) \notin \mathsf{Domain}(Map)$, so let us assume that $(ctxt, ad) \in \mathsf{Domain}(Map)$. Let $ctxt = (\mathcal{R}, \mathcal{V}, e, r'', c)$. By design, $\mathcal{R}$ and $\mathcal{S} = H_{\mathrm{dgd}}(ad, dc, ctxt)$ are column elements from the LOMDH challenger. By the same calculation we made in Game 5, we have

$$\mathcal{W}_i = \lambda^{(i)}\mathsf{DH}(\mathcal{R}, \mathcal{X}) + \sum_{j \in \boldsymbol{L}'} \lambda_j^{(i)} x_j \mathcal{R} + \sum_{j \in \boldsymbol{L}} \lambda_j^{(i)} z_j \mathcal{S} + \sum_{j \in \boldsymbol{L}' \setminus \boldsymbol{L}} \lambda_j^{(i)} \mathsf{DH}(\mathcal{S}, \mathcal{Z}_j).$$

So to compute $\mathcal{W}_i$, our LOMDH adversary makes an appropriate query to the LOMDH challenger to obtain

$$\lambda^{(i)}\mathsf{DH}(\mathcal{R}, \mathcal{X}) + \sum_{j \in \boldsymbol{L}' \setminus \boldsymbol{L}} \lambda_j^{(i)} \mathsf{DH}(\mathcal{S}, \mathcal{Z}_j). \tag{4}$$

At the end of the CCA attack game, our LOMDH adversary runs through all queries of the random oracle H_{kd}. For each such query $H_{\mathrm{kd}}(\mathcal{R}, \mathcal{U})$, of $\mathcal{R}$ is one of the column elements corresponding to an encryption query, the LOMDH adversary adds an pair to its output list corresponding to the relation $\mathcal{U} = \mathsf{DH}(\mathcal{R}, \mathcal{X})$.

We argue that the probability that our LOMDH adversary wins is precisely the probability that the failure event defined above occurs. This amounts to showing that each output relation $\mathcal{U} = \mathsf{DH}(\mathcal{R}, \mathcal{X})$ asserted by the LOMDH adversary is nontrivial. To see this, for each $i \in \{1, \ldots, N\} \setminus \boldsymbol{L}$, which may be used as in index in a decryption query, define the vector

$$\vec{v}_i \coloneqq \{\lambda_j^{(i)}\}_{j \in \boldsymbol{L}' \setminus \boldsymbol{L}} \in \mathbb{Z}_q^{\boldsymbol{L}' \setminus \boldsymbol{L}}.$$

By basic properties of polynomial interpolation, we see that any collection of at most $\Delta - 1$ of these vectors is linearly independent. Now consider a given encryption query that adds $(ctxt, ad)$ to $\mathsf{Domain}(Map)$, and let $ctxt = (\mathcal{R}, \ldots)$, so that $\mathcal{R}$ is a column element given by the LOMDH challenger. Further, consider a given decryption context dc, and let $\mathcal{S} = H_{\mathrm{dgd}}(ad, dc, ctxt)$, which is also a column element given by the LOMDH challenger. Then any collection of at most $\Delta - 1$

LOMDH query matrices, for queries of the form (4), cannot be linearly combined in such a way to zero out all the columns corresponding to the column elements $\mathcal{Z}_j$ for $j \in \boldsymbol{L}' \setminus \boldsymbol{L}$. This implies that all of the assertions made by our LOMDH adversary are nontrivial.

That completes the design and analysis of our LOMDH adversary. Note that in the LOMDH attack game, there will be at most $\Delta \leq t$ column elements; however, the number of row elements is only bounded by the number of encryption queries plus the number of queries to H_{dgd}, so there may be a significant (though still poly-bounded) number of these. Also, the number of pairs in the output list is bounded by the number of queries to H_{kd}. Note that each matrix submitted to the LOMDH adversary is actually very sparse, consisting of at most Δ nonzero elements.

That completes the proof of the theorem. □

6 A generic construction

In this section we show how to transform any (context-free) low-threshold CCA-secure decryption scheme into a context-dependent high-threshold CCA-secure decryption scheme. This resulting threshold decryption scheme has the same public key and encryption algorithm as the original scheme. This is be useful in legacy systems where the encryption algorithm cannot be changed, but where a context-dependent high-threshold CCA-secure decryption scheme is needed.

The key ingredient in our transformation is identity-based encryption (IBE) [51, 10]. Recall that an IBE scheme is a tuple of four algorithms $\mathsf{IBE} = (G, K, E, D)$, where $G() \to (mpk, msk)$ outputs the master public key mpk and master secret key msk; $K(msk, id) \to sk_{id}$ outputs the secret key for the identity id; $E(mpk, id, m) \to ctxt$ encrypts the message m for the identity id; and $D(sk_{id}, ctxt) \to m$ or `reject` decrypts the ciphertext $ctxt$ using the secret key sk_{id}.

In our settings we will need a *threshold* IBE scheme [10] where the master secret key msk is secret shared among N parties. The following definition reviews the syntax of such a scheme.

Definition 6. *A threshold IBE scheme defined over $(\boldsymbol{ID}, \boldsymbol{M})$ is a tuple of four efficient algorithms $\mathsf{IBE} = (G, K, E, D)$ where*

- *G is a probabilistic **key generation algorithm** that is invoked as*

$$(mpk, pkc, msk_1, \ldots, msk_N) \leftarrow\!\!\$\ G(N, t)$$

  *to generate the IBE master public key mpk and a t-out-of-N sharing of a master secret key msk. It outputs a **master public key** mpk, a **combiner key** pkc, and N **master key shares** $msk_1, \ldots, msk_N$.*

- *K is a (possibly) probabilistic **keygen algorithm** that is invoked as*

$$sk_{id,i} \leftarrow\!\!\$\ K(msk_i, id)$$

  where msk_i is one of the master key shares output by G, $id \in \boldsymbol{ID}$ is the identity of the requested secret key, and $sk_{id,i}$ is a secret key share for sk_{id} generated using msk_i.

- *E is a probabilistic **encryption algorithm** that is invoked as*

$$ctxt \leftarrow\!\!\$\ E(mpk, id, m),$$

  where mpk is a master public key output by G, id is an identity, and $m \in \boldsymbol{M}$ is a message.

- *D is a deterministic* ***decryption algorithm*** *that is invoked as*

$$m \leftarrow D(pkc, id, \boldsymbol{J}, \{sk_{id,j}\}_{j \in \boldsymbol{J}}, ctxt),$$

where pkc is a combiner key, $id \in \boldsymbol{ID}$ is an identity, $\boldsymbol{J}$ is a subset of $\{1, \ldots, N\}$ of size t, for $j \in \boldsymbol{J}$ the quantity $sk_{id,j}$ is a secret key share of sk_{id}, and $ctxt$ is a ciphertext. The algorithm either outputs a plaintext m, the special symbol `reject`*, or a special message* `blame`$(\boldsymbol{J}^*)$*, where $\boldsymbol{J}^*$ is a nonempty subset of $\boldsymbol{J}$.*

The correctness guarantee for IBE is that for every N, t, subset $\boldsymbol{J} \subset \{1, \ldots, N\}$ of size t, identity $id \in \boldsymbol{ID}$, and message m, it holds that

$$\Pr\left[D(pkc, id, \boldsymbol{J}, \{sk_{id,j}\}_{j \in \boldsymbol{J}}, ctxt) = m\right] = 1$$

where $(mpk, pkc, msk_1, \ldots, msk_N) \leftarrow\!\!\$\ G(N, t)$, $ctxt \leftarrow\!\!\$\ E(mpk, id, m)$, and $sk_{id,i} \leftarrow\!\!\$\ K(msk_i, id)$ for $i \in \boldsymbol{J}$.

We will define robustness and security for a threshold IBE scheme in Sections 6.2 and 6.3, respectively, after we present the transformation from (context-free) low-threshold decryptipm to context-dependent high-threshold decryption.

In the full version, we also present a construction of a threshold IBE scheme from the Boneh-Franklin IBE [10] satisfying all the necessary properties.

6.1 The construction

Let $\mathsf{IBE} = (G, K, E, D)$ be a threshold IBE scheme defined over $(\boldsymbol{ID}, \boldsymbol{M})$, as in Definition 6. Let $\mathsf{E} = (G, E, D, C)$ be a context-free threshold decryption scheme defined over $(\boldsymbol{M}, \boldsymbol{AD}, \boldsymbol{Ctxt})$, as in Remark 1.

We construct a derived context-dependent threshold decryption scheme, denoted $\mathsf{E}' = (G', E', D', C')$, with context space $\boldsymbol{DC}$. We assume that $\boldsymbol{Ctxt} \times \boldsymbol{AD} \times \boldsymbol{DC} \subseteq \boldsymbol{ID}$ so that a tuple $(ctxt, ad, dc)$ can be used as an IBE identity.

The threshold decryption scheme E' is presented in Fig. 1. As promised, the public key pk and the encryption algorithm E' are identical to the ones in the underlying threshold decryption scheme.

As presented, the key generation algorithm takes an additional parameter $t^\dagger$, which represents the reconstruction threshold for the low-threshold scheme E, and must satisfy $f < t^\dagger \leq t$. Moreover, it must be the case that $(N, f, t^\dagger)$ are allowable parameters for E. If E is a traditional threshold decryption scheme, then one would set $t^\dagger = f + 1$, but other parameter configurations are possible (see Remark 6).

This extra parameter does not match our definitions for the syntax, robustness, and security of threshold decryption. There are two ways of dealing with this issue. The first is to restrict ourselves to $t^\dagger = f + 1$, which likely covers the most important use cases. The second is to generalize our definitions so that in addition to the parameters (N, t, f), the key generation algorithm may take an additional, scheme-specific parameter (such as $t^\dagger$) which must satisfy certain constraints (such as $f < t^\dagger \leq t$). To be more general, we take the second approach. However, the reader may feel free to proceed assuming $t^\dagger = f + 1$.

- Algorithm $G'(N, t, f, t^\dagger)$ runs keygen for the threshold decryption scheme and IBE:

  1 : $(pk, pkc, sk_1, \ldots, sk_N) \leftarrow\!\$\ \mathsf{E}.G(N, t^\dagger, f)$
  2 : $(mpk, pkc', msk_1, \ldots, msk_N) \leftarrow\!\$\ \mathsf{IBE}.G(N, t)$
  3 : **output** $pk' \leftarrow pk, \quad pkc'' \leftarrow (pkc, pkc'), \quad sk'_i \leftarrow (sk_i, mpk, msk_i)$ for $i = 1, \ldots, N$.

- Algorithm $E'(pk', m, ad)$ is the same as for the threshold decryption scheme

  1 : **output** $\mathsf{E}.E(pk, m, ad)$

- Algorithm $D'(sk'_i, ctxt, ad, dc)$ uses $sk'_i = (sk_i, mpk, msk_i)$ and works as follows:

  1 : $ds_i \leftarrow\!\$\ \mathsf{E}.D(sk_i, ctxt, ad)$ // decrypt $ctxt$ using sk_i to get a decryption share
  2 : $adc \leftarrow (ctxt, ad, dc)$ // the augmented decryption context
  3 : $ctxt_i \leftarrow\!\$\ \mathsf{IBE}.E(mpk, adc, ds_i)$ // encrypt the share ds_i using adc as the identity
  4 : $sk_{dc,i} \leftarrow\!\$\ \mathsf{IBE}.K\big(msk_i, adc\big)$ // release one key share for the identity adc
  5 : **output** $ds'_i \coloneqq (dc, ctxt_i, sk_{dc,i})$

- Algorithm $C'\big(pkc'', ctxt, ad, dc, \boldsymbol{J}, \{ds'_j\}_{j\in\boldsymbol{J}}\big)$ uses $pkc'' = (pkc, pkc')$ and does:

  1 : **abort if** $dc \neq dc_j$ for some $j \in \boldsymbol{J}$ // recall that $ds'_j = (dc_j, ctxt_j, sk_{dc,j})$
  2 : $adc \leftarrow (ctxt, ad, dc)$ // the augmented decryption context
  3 : **for** $j \in \boldsymbol{J}$:
  4 : $ds_j \leftarrow \mathsf{IBE}.D(pkc', adc, \boldsymbol{J}, \{sk_{dc,i}\}_{i\in\boldsymbol{J}}, ctxt_j)$ // decrypt $ctxt_j$ using a quorum of key shares
  5 : **if** $ds_j = \texttt{blame}(\boldsymbol{J}^*)$
  6 : **then output** $\texttt{blame}(\boldsymbol{J}^*)$ **and exit**
  7 : **if** $ds_j = \texttt{reject}$
  8 : **then output** `reject` **and exit**
  9 : $\boldsymbol{J}^\dagger \leftarrow$ a canonical subset of $\boldsymbol{J}$ of size $t^\dagger$
  10 : $m \leftarrow \mathsf{E}.C(pkc, ctxt, ad, \boldsymbol{J}^\dagger, \{ds_j\}_{j\in\boldsymbol{J}^\dagger})$ // combine all the PKE decryption shares
  11 : **output** m

Figure 1: The context-dependent threshold scheme E' built from a context-free threshold scheme E and an IBE scheme IBE.

6.2 Robustness

In this section we prove the robustness of the derived threshold decryption scheme E' defined in Section 6.1. The scheme is robust if the underlying threshold decryption scheme E is robust, and in addition, the threshold IBE scheme IBE satisfies a notion of robustness that we now define.

The notion of robustness for threshold IBE is defined analogously to robustness for threshold decryption. Concretely, a robust threshold IBE scheme should satisfy two conditions: *accurate blaming* and *consistency*. Intuitively, accurate blaming requires that the decryption algorithm never blames parties due to honestly-generated secret key shares. It is defined using the following attack game.

Attack Game 4 (accurate blaming). *For a given IBE scheme* $\mathsf{IBE} = (G, K, E, D)$ *over* $(\boldsymbol{ID}, \boldsymbol{M})$ *and a given adversary* $\mathfrak{A}$*, we define the following attack game.*

- *The adversary sends to the challenger polynomially-bounded allowable parameters* (N, t)*.*
- *The challenger runs* $(mpk, pkc, msk_1, \ldots, msk_N) \leftarrow\!\!\$\ G(N, t)$ *and sends all this data to the adversary.*
- *The adversary sends to the challenger an index* $j^* \in \{1, \ldots, N\}$ *and an identity* $id \in \boldsymbol{ID}$*.*
- *The challenger runs* $sk \leftarrow\!\!\$\ K(msk_{j^*}, id)$ *and sends* sk *to the adversary.*
- *The adversary outputs*
$$(\boldsymbol{J}, \{sk_{id,j}\}_{j \in \boldsymbol{J}}, ctxt).$$
- *We say the adversary wins the game if*
  - $j^* \in \boldsymbol{J}$*.*
  - $sk_{id,j^*} = sk$*.*
  - $D(pkc, id, \boldsymbol{J}, \{sk_{id,j}\}_{j \in \boldsymbol{J}}, ctxt) = \mathtt{blame}(\boldsymbol{J}^*)$*, where* $j^* \in \boldsymbol{J}^*$*.*

We define $\mathfrak{A}$*'s advantage with respect to* E*, denoted* $\mathrm{blmIBEadv}[\mathfrak{A}, \mathsf{IBE}]$*, as the probability that* $\mathfrak{A}$ *wins the game.*

Definition 7 (accurate blaming). *We say that a threshold decryption scheme* E *provides* ***accurate blaming*** *if for all efficient adversaries* $\mathfrak{A}$*, the quantity* $\mathrm{blmIBEadv}[\mathfrak{A}, \mathsf{IBE}]$ *is negligible.*

The consistency property is also defined via a security game.

Attack Game 5 (consistent threshold IBE). *For a given threshold IBE scheme* $\mathsf{IBE} = (G, K, E, D)$ *over* $(\boldsymbol{ID}, \boldsymbol{M})$ *and a given adversary* $\mathfrak{A}$*, we define the following attack game.*

- *The adversary sends to the challenger polynomially-bounded allowable parameters* N *and* t*, where* $0 < t \leq N$*.*
- *The challenger runs* $(mpk, msk_1, \ldots, msk_N) \leftarrow\!\!\$\ G(N, t)$ *and sends all this data to the adversary.*

- *The adversary outputs*

$$(ctxt, id, \boldsymbol{J}_1, \{sk_{id,j}\}_{j\in\boldsymbol{J}_1}, \boldsymbol{J}_2, \{sk'_{id,j}\}_{j\in\boldsymbol{J}_2}),$$

  where $ctxt \in \boldsymbol{Ctxt}$, $id \in \boldsymbol{ID}$, $\boldsymbol{J}_1$ *and* $\boldsymbol{J}_2$ *are subsets of* $\{1,\ldots,N\}$ *of size* t, $\{sk_{id,j}\}_{j\in\boldsymbol{J}_1}$ *and* $\{sk'_{id,j}\}_{j\in\boldsymbol{J}_2}$ *are collections of secret key shares.*

- *We say the adversary wins the game if*
  - $D(id, \boldsymbol{J}_1, \{sk_{id,j}\}_{j\in\boldsymbol{J}_1}, ctxt) = m_1 \in \boldsymbol{M} \cup \{\texttt{reject}\}$,
  - $D(id, \boldsymbol{J}_2, \{sk'_{id,j}\}_{j\in\boldsymbol{J}_2}, ctxt) = m_2 \in \boldsymbol{M} \cup \{\texttt{reject}\}$, *and*
  - $m_1 \neq m_2$.

We define $\mathfrak{A}$*'s advantage with respect to* IBE*, denoted* conIBE$\mathsf{adv}[\mathfrak{A}, \mathsf{IBE}]$*, as the probability that* $\mathfrak{A}$ *wins the game.*

Definition 8 (consistent threshold IBE). *We say that a threshold IBE scheme* IBE *is* ***consistent*** *if for all efficient adversaries* $\mathfrak{A}$*, the quantity* conIBE$\mathsf{adv}[\mathfrak{A}, \mathsf{IBE}]$ *is negligible.*

Both the accurate blaming and the consistency properties can be strengthened to consider "perfect" robustness, requiring that blmIBE$\mathsf{adv}[\mathfrak{A}, \mathsf{IBE}]$ and/or conIBE$\mathsf{adv}[\mathfrak{A}, \mathsf{IBE}]$ are equal to 0 for any adversary $\mathfrak{A}$, respectively. We will prove that our generic construction satisfies computational accurate blaming and consistency. However, if E has perfect accurate blaming and IBE satisfies perfect accurate blaming and consistency, then the proof can be easily extended to show that the resulting high-threshold context-dependent threshold decryption scheme E′ has perfect accurate blaming as well.

In our proofs, we will implicitly assume that $t^\dagger = f+1$ to syntactically match Definitions 2 and 3, but both these definitions and our proofs readily extend to accommodate this extra parameter.

Equipped with the above definitions, we now prove that our generic construction satisfies both robustness properties defined in Section 2.1.1. We start, in Lemma 1, by proving that it satisfies accurate blaming.

Lemma 1. *If* E *satisfies accurate blaming per Definition 2 and* IBE *satisfies accurate blaming and consistency per Definitions 7 and 8, respectively, then* E′ *satisfies accurate blaming per Definition 2.*

Proof. Let $\mathfrak{A}$ be an adversary participating in the accurate blaming attack game of E (Attack Game 1). We claim that there are adversaries $\mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_3$ such that

$$\text{blmPKE}\mathsf{adv}[\mathfrak{A}, \mathsf{E}'] \leq \text{blmIBE}\mathsf{adv}[\mathfrak{B}_1, \mathsf{IBE}] + \text{conIBE}\mathsf{adv}[\mathfrak{B}_2, \mathsf{IBE}] + \text{blmPKE}\mathsf{adv}[\mathfrak{B}_3, \mathsf{E}],$$

and the lemma follows.

To see why, it will be convenient to explicitly recall Attack Game 1 for E′ with the adversary $\mathfrak{A}$. At the beginning of the game, $\mathfrak{A}$ sends parameters N, t and f. In response, the challenger runs $(pk, pkc, sk_1, \ldots, sk_N) \leftarrow\!\$\ \mathsf{E}.G(N, t, f)$ and $(mpk, pkc', msk_1, \ldots, msk_N) \leftarrow\!\$\ \mathsf{IBE}.G(N, t)$, and sends $pk' \leftarrow pk$, $pkc'' \leftarrow (pkc, pkc')$, and $sk'_i \leftarrow (sk_i, mpk, msk_i)$ for $i = 1, \ldots, N$ to the adversary. The adversary $\mathfrak{A}$ then queries the challenger on an index $j^* \in \{1, \ldots, N\}$, a ciphertext $ctxt$, associated data ad, and decryption context dc. In response, the challenger computes $ds \leftarrow\!\$\ \mathsf{E}.D(sk_{j^*}, ctxt, ad)$ and $ctxt_{j^*} \leftarrow\!\$\ \mathsf{IBE}.E(mpk, adc, ds)$, where $adc = (ctxt, ad, dc)$. The challenger then runs $sk \leftarrow\!\$\ \mathsf{IBE}.K(msk_{j^*}, adc)$ and returns $ds' \coloneqq (dc, ctxt_{j^*}, sk)$ to $\mathfrak{A}$.

Finally, the adversary outputs

$$(\boldsymbol{J}, \{ds'_j\}_{j\in\boldsymbol{J}}),$$

where for every $j \in \boldsymbol{J}$, $ds'_j := (dc^*, ctxt_j, sk_j)$. The adversary wins if $j^* \in \boldsymbol{J}$, $ds'_{j^*} = ds'$, and $\mathsf{E}'.C(pkc, ctxt, ad, \boldsymbol{J}, \{ds'_j\}_{j\in\boldsymbol{J}}, dc) = \mathtt{blame}(\boldsymbol{J}^*)$, where $j^* \in \boldsymbol{J}^*$. Suppose that these three conditions are satisfied.

By construction, C' outputs $\mathtt{blame}(\boldsymbol{J}^*)$ either in Step 6 as the output of $\mathsf{IBE}.D$, or in Step 11 as the output of $\mathsf{E}.C$. Suppose C' outputs $\mathtt{blame}(\boldsymbol{J}^*)$ in Step 6, and call this event $\mathsf{Blm1}$. This case immediately reduces to the accurate blaming of IBE, since the winning condition $ds'_{j^*} = ds'$ implies in particular that $sk_j = sk$ and sk was honestly generated by the challenger as the output of $\mathsf{IBE}.K(msk_{j^*}, adc)$. Therefore, there exists an adversary $\mathfrak{B}_1$ that wins the accurate blaming game for IBE whenever $\mathsf{Blm1}$ occurs.

Now suppose that C' outputs $\mathtt{blame}(\boldsymbol{J}^*)$ in Step 11 and call this event $\mathsf{Blm2}$. Let $\underline{ds}_{j^*}$ be the result of decrypting the IBE-ciphertext $ctxt_{j^*}$ in the partial decryption computed by the challenger, using the key shares $\{ds'_j\}_{j\in\boldsymbol{J}}$ outputted by the adversary. That is, let $\underline{ds}_{j^*} \leftarrow \mathsf{IBE}.D(pkc', adc, \boldsymbol{J}, \{sk_j\}_{i\in\boldsymbol{J}}, ctxt_{j^*})$. We consider a partition of $\mathsf{Blm2}$ to two complementing events:

- When $\underline{ds}_{j^*} \neq ds$ occurs, we construct an adversary $\mathfrak{B}_2$ breaking the consistency of IBE. The adversary $\mathfrak{B}_2$ invokes $\mathfrak{A}$ and simulates Attack Game 1 to it in the natural way, using the IBE keys that it gets from the challenger in the IBE consistency game and sampling the keys to E on its own. When $\mathfrak{A}$ outputs $(\boldsymbol{J}, \{ds'_j\}_{j\in\boldsymbol{J}})$, $\mathfrak{B}_2$ computes honestly-generated key shares for adc for every $i \in \boldsymbol{J}$ by running $\widehat{sk}_j \leftarrow\!\!\$\ \mathsf{IBE}.K(msk_j, adc^*)$ and then computes the honest decryption of $ctxt_{j^*}$ with these key shares by running $\widehat{ds}_{j^*} \leftarrow \mathsf{IBE}.D(pkc', adc, \boldsymbol{J}, \{\widehat{sk}_j\}_{i\in\boldsymbol{J}}, ctxt_{j^*})$. By the correctness of IBE, it must hold that $\widehat{ds}_{j^*} = ds$, implying $\widehat{ds}_{j^*} \neq \underline{ds}_{j^*}$. Therefore, by outputting

  $$(ctxt_{j^*}, id = adc, \boldsymbol{J}_1 = \boldsymbol{J}, \{sk_j\}_{j\in\boldsymbol{J}}, \boldsymbol{J}_2 = \boldsymbol{J}, \{\widehat{sk}_j\}_{j\in\boldsymbol{J}}),$$

  $\mathfrak{B}_2$ wins the consistency game of IBE.

- If $\underline{ds}_{j^*} = ds$, then we can construct an adversary $\mathfrak{B}_3$ breaking the accurate blaming of E. The adversary $\mathfrak{B}_3$ invokes $\mathfrak{A}$ and simulates Attack Game 1 to it in the natural way, using the E keys that it gets from the challenger in the threshold decryption accurate blaming game and sampling the keys to IBE on its own. It also forwards the partial decryption query of $\mathfrak{A}$ to its own challenger, and then uses the response and knowledge of the IBE keys to respond to $\mathfrak{A}$'s queries. When $\mathfrak{A}$ returns an output $(\boldsymbol{J}, \{ds'_j\}_{j\in\boldsymbol{J}}$ where $ds'_j := (dc, ctxt_j, sk_j)$ for $j \in \boldsymbol{J}$, $\mathfrak{B}_3$ outputs

  $$(\boldsymbol{J}, \{\underline{ds}_j\}_{j\in\boldsymbol{J}}),$$

  where $\underline{ds}_j := \mathsf{IBE}.D(pkc', adc, \boldsymbol{J}, \{sk_j\}_{i\in\boldsymbol{J}}, ctxt_j)$ for every $j \in \boldsymbol{J}$. Since $\mathfrak{A}$ wins in the accurate blaming game for E', it holds that $j^* \in \boldsymbol{J} \cap \boldsymbol{J}^*$. Moreover, we are in the case where $\underline{ds}_{j^*} = ds$, it holds that $\mathfrak{B}_3$ also wins the accurate blaming for E.

Overall, we have that $\mathrm{conIBEadv}[\mathfrak{B}_2, \mathsf{IBE}] + \mathrm{blmPKEadv}[\mathfrak{B}_3, \mathsf{E}] = \Pr[\mathsf{Blm2}]$, and the lemma follows. □

Lemma 2 below proves that our generic construction satisfies consistency.

Lemma 2. *If* E *satisfies consistency per Definition 3, then* E' *satisfies consistency per Definition 3.*

Proof. Let $\mathfrak{A}$ be an adversary taking part in the consistency game of E' (Attack Game 2). We show that there exists an adversary $\mathfrak{B}$ taking part in consistency game of E for which

$$\text{conPKEadv}[\mathfrak{B}, \mathsf{E}] = \text{conPKEadv}[\mathfrak{A}, \mathsf{E}'],$$

and then the consistency of E' follows from the consistency of E. The adversary $\mathfrak{B}$ is defined as follows:

1. Invoke $\mathfrak{A}$ and get parameters $(N, t, f, t^\dagger)$. Forward these parameters to the challenger.
2. Receive $(pk, pkc, sk_1, \ldots, sk_N)$ from the challenger. Sample $(mpk, pkc', msk_1, \ldots, msk_N) \leftarrow\!\!\$\ \mathsf{IBE}.G(N, t)$ and send $(pk', pkc'', sk'_1, \ldots, sk'_N)$ to $\mathfrak{A}$, where $pk' = pk$, $pkc' = (pkc, pkc')$, and $sk'_i = (sk_i, mpk, msk_i)$ for $i = 1, \ldots, N$.
3. $\mathfrak{A}$ outputs $(ctxt, ad, \boldsymbol{J}_1, \{ds_{1j}\}'_{j\in\boldsymbol{J}_1}, dc_1, \boldsymbol{J}_2, \{ds_{2j}\}'_{j\in\boldsymbol{J}_2}, dc_2)$, where $ds'_{1j} = (dc_1, ctxt_{1j}, sk_{1,dc,j})$ for $j \in \boldsymbol{J}_1$ and $ds'_{2j} = (dc_2, ctxt_{2j}, sk_{2,dc,j})$ for $j \in \boldsymbol{J}_2$. For $i \in \{1, 2\}$ and $j \in \boldsymbol{J}_i$, $\mathfrak{B}$ computes $ds_{i,j} \leftarrow \mathsf{IBE}.D(pkc', adc, \boldsymbol{J}_i, \{sk_{i,dc,j}\}_{j\in\boldsymbol{J}_i}, ctxt_{ij})$. If any of these invocation results in an output $\texttt{blame}(\boldsymbol{J}^*)$, then $\mathfrak{B}$ aborts. Otherwise, $\mathfrak{B}$ computes $m_i \leftarrow \mathsf{E}.C(pkc, ctxt, ad, \boldsymbol{J}_i^\dagger, \{ds_{i,j}\}_{j\in\boldsymbol{J}_i^\dagger})$ for $i \in \{1, 2\}$ and outputs $(ctxt, ad, \boldsymbol{J}_1^\dagger, \{ds_{1j}\}_{j\in\boldsymbol{J}_1^\dagger}, dc_1, \boldsymbol{J}_2^\dagger, \{ds_{2j}\}_{j\in\boldsymbol{J}_2^\dagger}, dc_2)$. Here, for $i \in \{1, 2\}$, the set $\boldsymbol{J}_i^\dagger$ is the canonical subset of size $t^\dagger$ of $\boldsymbol{J}$ as computed by C'.

Observe that whenever $\mathfrak{A}$ wins its game, so does $\mathfrak{B}$, and the lemma follows. □

6.3 Security

We next prove security of the derived threshold decryption scheme E' defined in Section 6.1. The scheme is high-threshold CCA secure if the underlying threshold decryption scheme E is low-threshold CCA secure and the IBE scheme IBE is semantically secure. We have already defined the notion of low-threshold CCA security for E. Let us now define the notion of semantic security we will require for IBE. As usual, this is done via a game.

Attack Game 6 (IBE Semantic Security). *For a threshold IBE scheme $\mathsf{IBE} = (G, K, E, D)$, defined over $(\boldsymbol{ID}, \boldsymbol{M})$, and for a given adversary $\mathfrak{A}$, we define two experiments.*

Experiment b $(b = 0, 1)$**:**

- Setup: *the adversary sends polynomially-bounded allowable parameters N and t, where $0 < t \le N$, and a subset $\boldsymbol{L} \subseteq \{1, \ldots, N\}$, where $\Delta := t - |\boldsymbol{L}| > 0$, to the challenger. The challenger computes $(mpk, pkc, msk_1, \ldots, msk_N) \leftarrow\!\!\$\ G(N, t)$, and sends mpk, pkc, and $\{msk_\ell\}_{\ell\in\boldsymbol{L}}$ to the adversary.*
- $\mathfrak{A}$ *then makes a series of queries to the challenger. Each query is one of two types:*
  - Key query: *each key query consists of a pair $(id, i) \in \boldsymbol{ID} \times \{1, \ldots, N\} \setminus \boldsymbol{L}$. The challenger computes the secret key $sk \leftarrow K(msk_i, id)$, and sends sk to $\mathfrak{A}$.*
  - Encryption query: *a query consists of a triple $(id, m_0, m_1) \in \boldsymbol{ID} \times \boldsymbol{M}^2$, where m_0 and m_1 are messages of the same length. The challenger computes $ctxt \leftarrow\!\!\$\ E(mpk, id, m_b)$ and sends ctxt to $\mathfrak{A}$.*

We require that for every identity used in an encryption query, the adversary issues at most $\Delta - 1$ key queries for that identity.

- *At the end of the game, $\mathfrak{A}$ outputs a bit $\bar{b} \in \{0,1\}$.*

*If W_b is the event that $\mathfrak{A}$ outputs 1 in Experiment b, define $\mathfrak{A}$'s **advantage** with respect to* IBE *as*

$$\mathrm{SSadv}[\mathfrak{A}, \mathsf{IBE}] := \Big|\Pr[W_0] - \Pr[W_1]\Big|.$$

Definition 9 (IBE Semantic Security). *A threshold IBE scheme is **semantically secure** if for all efficient adversaries $\mathfrak{A}$, the value* $\mathrm{SSadv}[\mathfrak{A}, \mathsf{IBE}]$ *is negligible.*

We can now prove security of the construction E' from Section 6.1. This is captured in the following theorem.

Theorem 3. *If* E *is (context-free) low-threshold CCA secure and* IBE *is semantically secure, then* E' *is a context-dependent high-threshold CCA secure.*

Proof. Let $\mathfrak{A}$ be an efficient adversary attacking E' in the high-threshold CCA security attack game. We prove that there exist efficient adversaries $\mathfrak{B}$ and $\mathfrak{C}$ such that

$$\mathrm{hthCCAadv}^*[\mathfrak{A}, \mathsf{E}'] \leq \mathrm{SSadv}[\mathfrak{B}, \mathsf{IBE}] + \mathrm{lthCCAadv}^*[\mathfrak{C}, \mathsf{E}] + \mathsf{Coll}_{\mathfrak{A}}. \tag{5}$$

Here, we are using the bit-guessing advantages for the low-threshold and high-threshold decryption schemes E and E', as discussed in Remark 7. In addition, $\mathsf{Coll}_{\mathfrak{A}}$ is a certain ciphertext collision probability, which we will show is negligible under the assumption that E is low-threshold CCA secure (in fact, just semantically secure).

We present a sequence of games, Game 0, Game 1, Game 2. In each Game j, p_j denotes the adversary's guessing probability, that is, the probability that the adversary correctly guesses the hidden bit b of the challenger.

Game 0. This is the bit-guessing version of Attack Game 3, for the encryption scheme E' and adversary $\mathfrak{A}$. Since future games will change how encryption and decryption queries are answered, we explicitly write how they are answered in this game:

- Encryption queries: On input (m_0, m_1, ad), the challenger computes $ctxt \leftarrow\!\$\ \mathsf{E}.E(pk, m_b, ad)$ and replies with $ctxt$. If $(ctxt, ad) \notin \mathsf{Domain}(Map)$, then it also initializes $Map[ctxt, ad] \leftarrow \emptyset$.
- Decryption queries: On input $(ctxt, ad, dc, i)$ that meets either the precondition (D1) $\vee$ (D2), the challenger computes the E-decryption share $ds_i \leftarrow\!\$\ \mathsf{E}.D(sk_i, ctxt, ad, dc)$, sets the augmented decryption context $adc \leftarrow (ctxt, ad, dc)$, encrypts $ctxt_i \leftarrow\!\$\ \mathsf{IBE}.E(mpk, adc, ds_i)$, and derives the identity key $sk_{dc,i} \leftarrow\!\$\ \mathsf{IBE}.K\big(msk_i, adc\big)$. It then replies to $\mathfrak{A}$ with $ds'_i := (dc, ctxt_i, sk_{dc,i})$. If $(ctxt, ad) \in \mathsf{Domain}(Map)$, then the challenger also updates $Map[ctxt, ad] \leftarrow Map[ctxt, ad] \cup \{(dc, i)\}$.

Game 1. In this game, we modify the adversary so that if a ciphertext/associated-data pair $(ctxt, ad)$ associated with an encryption query ever matches that of a previous decryption query, then the adversary immediate aborts. We have

$$|p_1 - p_0| \leq \mathsf{Coll}_{\mathfrak{A}}, \tag{6}$$

where $\mathsf{Coll}_{\mathfrak{A}}$ is the probability that this "collision abort" rule is triggered in Game 1. We know that $\mathsf{Coll}_{\mathfrak{A}}$ is negligible assuming E is low-threshold CCA secure. This is a standard fact that holds for any semantically secure public-key encryption scheme. In fact, for most schemes (such as those in [54]), this holds unconditionally, as the encryption algorithm produces ciphertexts with very high entropy.

Game 2. In this game, we modify how decryption queries are answered. On a query of the form $(ctxt, ad, dc, i) \in \boldsymbol{Ctxt} \times \boldsymbol{AD} \times \boldsymbol{DC} \times \{1, \ldots, N\} \setminus \boldsymbol{L}$, the challenger first checks if $(ctxt, ad) \in \mathsf{Domain}(Map)$. If not, then it answers the query as in Game 0. If $(ctxt, ad) \in \mathsf{Domain}(Map)$, then the challenger computes $ctxt_i \leftarrow\!\$\ \mathsf{IBE}.E(mpk, adc, 0)$ instead of $ctxt_i \leftarrow\!\$\ \mathsf{IBE}.E(mpk, adc, ds_i)$. That is, the challenger encrypts the message 0 to identity adc using IBE, instead of encrypting ds_i.

It is fairly straightforward to see that $|p_2 - p_1|$ is negligible assuming IBE is semantically secure. Specifically, there exists an adversary $\mathfrak{B}$, with essentially the same running time as $\mathfrak{A}$, such that

$$|p_2 - p_1| = \mathrm{SSadv}[\mathfrak{B}, \mathsf{IBE}]. \tag{7}$$

Adversary $\mathfrak{B}$ runs in the obvious way, running $\mathfrak{A}$, but supplying its challenger triples $(adc, ds_i, 0)$, and outputting whatever $\mathfrak{A}$ outputs. We note that the point of introducing the "collision abort" in Game 1 was to ensure the decryption precondition (D1) $\vee$ (D2) enforced in the Attack Game 3 implies that in Attack Game 6, the requirement that "for every identity used in an encryption query, the adversary issues at most $\Delta - 1$ key queries for that identity" is always met.

It also is fairly straightforward to see that $|p_2 - 1/2|$ is negligible assuming E is low-threshold CCA secure. Specifically, there exists an adversary $\mathfrak{C}$, with essentially the same running time as $\mathfrak{A}$, such that

$$|p_2 - 1/2| = \mathrm{lthCCAadv}[\mathfrak{C}, \mathsf{E}]. \tag{8}$$

Adversary $\mathfrak{C}$ runs in the obvious way, exploiting the fact that in Game 2, when processing decryption queries, the challenger only needs to decrypt ciphertexts when the more restrictive precondition (D1) holds, in accordance with the rules of the low-threshold decryption attack game.

(5) follows immediately from (6), (7), and (8). □

7 Applications

7.1 Encrypted atomic broadcast

A core primitive underlying blockchain is **atomic broadcast**, which allows a group of parties to agree on a sequence of transactions submitted by clients. An adversary who controls the network and some of the parties can censor and re-order these transactions, potentially obtaining monetary gains at the expense of other clients. To effectively carry out such an attack, the adversary typically needs to know the contents of the transactions submitted by these other clients. Thus, one way to mitigate against such an attack is to allow clients to submit encrypted transactions. Such an encrypted transaction should only be decrypted after it has been added to the sequence of agreed upon transactions.

To implement this idea, one could use a public-key encryption scheme that supports threshold decryption, so that parties release their decryption share of an encrypted transaction only after that transaction has been added to the sequence of agreed upon transactions. Unfortunately, this simple approach adds an extra round of communication to the protocol. The goal of this section is to show

how to use a context-dependent high-threshold decryption scheme to do this without incurring an extra round of communication. We focus on ubiquitous *leader-based* atomic broadcast protocols that operate in the *partially synchronous* model, such as PBFT [25], HotStuff [43], Tendermint [19], ICC [21], and Simplex [26].

To this end, we first present a simple framework that allows us to capture the aspects of these (and other) protocols that are relevant to our particular task, while ignoring (the many) details that are not relevant. Then we show how we can "piggyback" the transmission of decryption shares with other protocol messages, so that transmitting these decryption shares does not increase the latency in the optimistic case (where the leader is honest and the network is synchronous). As we will see, using a CCA-secure context-dependent high-threshold decryption scheme, provides the confidentiality property we want. We will also see that using a high-threshold for decryption is not enough by itself — to achieve confidentiality, we need to use the decryption context mechanism in a particular way.

7.1.1 A simple framework for a general class of atomic broadcast protocols.

As mentioned above, we first present a simple framework that allows us to capture the aspects of several atomic broadcast protocols that are relevant to our particular task, while ignoring details that are not relevant. This framework applies to a number of protocols, including PBFT, HotStuff, Tendermint, ICC, and Simplex. We stress that our goal here is not to fully describe and analyze these protocols, but rather, to identify the common characteristics of these protocols that are relevant to our task. The reader who is familiar with any one of these protocols should be able to easily convince themselves that this framework applies to that protocol. We assume that there are n parties running the protocol, and that f is a bound on the number of these parties that may be corrupt. We denote by f' the number of parties that are actually corrupt in a given execution of the protocol.

In this framework, an atomic broadcast protocol proceeds by building a (directed) *tree of locked blocks*. Each such block is a node in this tree, and has a unique parent. We assume a canonical genesis block that is the root of the tree. Each block also contains a *payload*, which is a sequence of transactions, some of which may be encrypted. The same transaction may appear in more than one block. At any given point in time, the parties running the protocol may have somewhat different views of the tree of locked blocks; however, each such view will be a subtree of this tree also rooted at the same genesis block.

Within this tree of locked blocks, the protocol identifies a *path of committed blocks*. At any given point in time, the parties running the protocol may have somewhat different views of the path of committed blocks. However, each such view will be a subpath of this path, also rooted at the genesis block. The ordered sequence of transactions produced by the protocol is derived from the payloads of the blocks along the path of committed blocks.

Now, different protocols may use different mechanisms for building the tree of locked blocks and identifying the path of committed blocks, and we shall not concern ourselves with the details of these mechanisms, except as follows. Namely, we assume that a voting mechanism is used to identify committed blocks. Specifically, at various points in time, a party may cast a *commit vote* for a block in its view of the tree of locked blocks. The exact logic by which the protocol uses these commit votes to identify committed blocks is not relevant, except that we require that the following condition holds:

Commit Inclusion Condition: *if at least $n - f - f'$ honest parties vote to commit a block B, then B must eventually be included on the path of committed blocks.*

This condition is typically used to prove the safety property of the atomic broadcast protocol. How this condition is enforced depends on the specific logic of the protocol.

7.1.2 Piggybacking decryption shares.

We assume that encrypted transactions are encrypted using a CCA-secure context-dependent threshold decryption scheme with decryption threshold $t \coloneqq n - f$. We assume that we can derive a decryption context from any given block in the tree of locked blocks in such a way that different blocks in the tree yield different decryption contexts. This can easily be achieved by deriving the decryption context for a given block by hashing its payload and any other necessary data to distinguish that block from other blocks in the tree. For many protocols, such a decryption context is readily at hand in the form of a "block hash".

We then augment the atomic broadcast protocol as follows: for a given block B in the tree of locked blocks, a party will broadcast its decryption share of every encrypted transaction in the payload of B using the decryption context derived from B, when:

- that party casts a commit vote for a block B, or
- that party identifies B as a committed block.

In addition, whenever a party has identified a block B as a committed block, it will wait for $n - f$ corresponding decryption shares to decrypt any and all encrypted shares in the payload of B.

Confidentiality. It should be clear that from the security properties of the threshold decryption scheme, and the Commit Inclusion Condition, we have the following confidentiality property:

> *if an adversary learns anything about the plaintexts of the encrypted transactions in the payload of a block B, then B must eventually be included on the path of committed blocks.*

To see this in more detail, suppose the adversary obtains enough decryption shares to obtain any information about the plaintexts of any encrypted transactions in a block B. By the security properties of the threshold decryption scheme, the adversary must have obtained at least $t - f' = n - f - f'$ such shares from honest parties, with the same decryption context. So consider the point in time when the last such share was generated by an honest party. Suppose that at this time, some honest party has already identified B as a committed block. Then we know that B is already on the path of committed blocks. Otherwise, all of these $n - f - f'$ shares must have been piggybacked with commit votes for B. But then, by the Commit Inclusion Condition, that B must eventually be included on the path of committed blocks.

The need for a decryption context. We stress that the use of a block-specific decryption context is essential to realize the confidentiality property. This is because the same encrypted transaction c may be included in different payloads of blocks in the tree of locked blocks. If we piggyback decryption shares for c as above, but do not use decryption contexts, then an adversary could obtain a few decryption shares for c from one block B_1, and a few more from another block

B_2, so that these shares can be aggregated together to decrypt c, and yet neither B_1 nor B_2 will ever be included in the path of finalized blocks. However, with decryption contexts, the adversary cannot aggregate decryption shares from disparate blocks in this way. That is the key feature of decryption contexts that is essential here.

Latency. This piggybacking technique can increase the latency of the protocol by one round of communication in the worst case. However, the latency will typically not increase at all in optimistic case. This is certainly true for the protocols PBFT, HotStuff, Tendermint, ICC, and Simplex, as the reader familiar with any of these protocols may easily verify.

7.2 Encrypted auction systems

To illustrate the versatility of context-dependent threshold decryption we describe one more application, this time in the context of sealed-bid auctions. For brevity we will keep the discussion informal.

Consider an encryption-based sealed-bid auction system. Bidders submit encrypted bids, encrypted with respect to some public key pk. The corresponding secret decryption key is shared among N trusties so that t decryption shares are needed to decrypt a given encrypted bid. Bidders submit their encrypted bids by posting them on a public bulletin board, and bids are accepted up to some preset deadline T. After time T the trustees post on the bulletin board the decryption shares for all submitted bids. Anyone can then decrypt the set of submitted bids and determine the winner. The system is not meant to provide privacy for losing bids; the intent is that all submitted bids will be made public once the auction is over. We assume that the bulletin board is append-only and censorship resistant so that bids cannot be censored or changed after they are posted.

Here we focus on the sealed-bid aspect of the system. In particular, let us modify the mechanism a bit so that the deadline T is chosen dynamically based on certain conditions, such as the contents of the bulletin board, or some external source of randomness. For example, in a *candle auction* the deadline T is sampled dynamically from a distribution so that bidders do not know ahead of time the exact time when the auction will end (historically, the auction ended when a candle flame expired, hence the name). This is intended to prevent all the bids from coming in at the last minute.

Now, suppose the auction system is implemented using a context-free threshold decryption scheme. Suppose that at time T_1 some set of $t/2$ trustees are fooled into believing that the auction is over. Consequently, they incorrectly post their decryption shares to the bulletin board. Shortly after, they realize their mistake, but they can no longer remove the decryption shares from the bulletin board. Later, at time $T_2 > T_1$, some other set of $t/2$ trustees are fooled into believing that the auction is over. They also post their decryption shares to the bulletin board. With a context-free threshold decryption scheme, this will cause all bids to become available in the clear. The problem is that neither at time T_1 nor time T_2 was there a quorum of trustees who believed that the auction was over, and yet the bids are decrypted.

This problem can be easily avoided using a context-dependent scheme. When the trustees compute a decryption share, they use their perceived deadline T as the decryption context. Now the bids will be decrypted only when t or more trustees believe that the auction ended at time T. The bids remain secure even if the problematic situation described in the previous paragraph takes place.

8 Stronger notions of security

In this section, we briefly sketch stronger definitions of CCA security than that presented in Section 2.1.2. We first consider adaptive corruptions. Then we give a simulation based definition.

8.1 Adaptive corruptions

Attack Game 3 models static corruptions in that the adversary must select the set $\boldsymbol{L}$ of corrupt parties at the very beginning of the attack game. We can modify the game to model adaptive corruptions as follows.

The setup stage is exactly as in Attack Game 3. In particular, the adversary specifies a set of corrupt parties $\boldsymbol{L}$ in the setup stage. In addition, we add a new type of query called a *corruption* query, in which the adversary specifies the index ℓ of an honest party to corrupt. The challenger adds ℓ to the set $\boldsymbol{L}$ and gives sk_ℓ to the adversary. For low-threshold security, such a corruption query is only allowed if it would not make the size of $\boldsymbol{L}$ exceed the corruption bound f. For high-threshold security, corruption queries must also satisfy another precondition, which we specify in (9) below.

Encryption queries are processed just as in Attack Game 3. Decryption queries are processed just as in Attack Game 3, except that for high-threshold security, the precondition (D2) will be modified, as specified in (9) below. For low-threshold security the adversary cannot issue decryption queries for any of the challenge ciphertexts it obtains in response to encryption queries, or more precisely, precondition (D2) in Attack Game 3 is deleted, as in the static corruption setting.

As mentioned above, for high-threshold security, corruption and decryption queries must satisfy a certain precondition. This precondition is that processing the query would not cause the following condition to hold:

for some $(ctxt, ad) \in \mathsf{Domain}(Map)$ *and for some* dc*, we have*

$$|\boldsymbol{L}| + \big|\{j \in [N] \setminus \boldsymbol{L} : (dc, j) \in Map[ctxt, ad]\}\big| \geq t. \tag{9}$$

We do not claim that the schemes presented in this paper are secure against adaptive corruptions. Our proofs of security under the stated assumptions cannot be easily modified to prove security against adaptive corruptions. However, it seems plausible that these schemes, as presented, may be secure against adaptive corruption under stronger (but still reasonable) assumptions.

Context-free low vs. high threshold security. When considering adaptive corruptions in the context-free setting (i.e., when the context space $\boldsymbol{DC}$ contains at most one element), we observe that low-threshold security and high-threshold security are equivalent notions. This is captured in the following theorem.

Theorem 4. *Let* E *be a context-free threshold decryption scheme, and let us require that* $f = t - 1$ *in Attack Game 3. Then if* E *is low-threshold CCA secure under adaptive corruptions, it is also high-threshold CCA secure under adaptive corruptions.*

Proof. We wish to show that for every adversary $\mathfrak{A}$ that breaks the high-threshold security of E there is an adversary $\mathfrak{B}$ that breaks its low-threshold security with adaptive corruptions. To that end, we consider modified attack games, in which the adversary is restricted to issuing a single

encryption (challenge) query. This notion is referred to as 1CCA security in [16, Ch. 12]. It is well-known that this change incurs a multiplicative loss in security that is bounded by the number of encryption queries issued by the adversary; in particular, it is polynomially-bounded for polynomial-time adversaries. Boneh and Shoup [16, Th. 12.1] gave a proof in the setting of CCA-secure public key encryption, which readily extends to the threshold case as well.[3]

Now, let $\mathfrak{A}$ be an adversary for the high-threshold setting as above. The adversary $\mathfrak{B}$ is defined similarly to $\mathfrak{A}$ with one modification: after $\mathfrak{A}$ issues its single encryption query with associated data ad^* and obtains a challenge ciphertext $ctxt^*$, it may issue decryption queries with $ctxt^*$ and ad^* to obtain partial decryptions (as it is a high-threshold adversary). Instead, whenever $\mathfrak{A}$ issues a query of the form $(ctxt^*, ad^*, i)$, the low-threshold adversary instead issues a corruption query for party i. Then, using sk_i, $\mathfrak{B}$ can compute the response to the query $(ctxt^*, ad^*, i)$ on its own. Note that $\mathfrak{B}$ never issues a decryption query of the form $(ctxt^*, ad^*, i)$ after getting the challenge ciphertext. Hence, whenever $\mathfrak{A}$ breaks the high-threshold security of E, $\mathfrak{B}$ breaks its low-threshold security. □

Theorem 4 is specific to the setting of **context-free** threshold decryption and **adaptive corruptions**. Both of these restrictions seem necessary:

- The adversary $\mathfrak{B}$ from the proof corrupts parties adaptively, after observing the challenge ciphertext. Hence, it is not an admissible adversary for the static corruptions attack game, as defined in Section 2.1.2. It is not hard to come up with examples of threshold decryption schemes that are low-threshold secure but are not high-threshold secure in the static corruptions setting.

- The notions of adaptive low-threshold and adaptive high-threshold security are also different when considering context-dependent threshold decryption. The reason is that with a decryption context, the adversary can obtain decryption shares for the same challenge ciphertext from more than t parties. For example, for some $u > 1$ it can obtain $u \cdot (t-1)$ decryption shares from $u \cdot (t-1)$ distinct parties under u different decryption contexts. These decryption queries cannot be reduced by the adversary $\mathfrak{B}$ to corruption queries since this would require $u \cdot (t-1)$ key corruptions, which is forbidden by the attack game. In fact, any CCA-secure threshold decryption scheme that simply ignores the context is an example that separates high-threshold security from low-threshold security in the context-dependent setting. For such a scheme, any t decryption shares for the challenge ciphertext using different decryption contexts suffice to decrypt the challenge ciphertext, so the scheme is not high-threshold secure. On the other hand, the scheme is context-dependent low-threshold secure since, as discussed in Section 1, context dependence has no effect in the low-threshold setting: any context-free CCA-secure threshold decryption is also context-dependent low-threshold secure as well because the adversary cannot issue decryption queries for the challenge ciphertext.

8.2 Simulation CCA security

We now sketch an alternative to the game-based definition of security in Section 2.1.2. This definition is a simulation-based definition that models threshold decryption as a *service*, and which may make it much easier to analyze systems that use threshold decryption. The definition presented here

[3]The one difference is that in the high-threshold setting, the reduction also has to forward partial decryption queries for the challenge ciphertext to its own challenger.

is closely related to that in [23], but with some differences and generalizes. Our definition is general enough to allow us to model any combination of high/low-threshold security and static/adaptive corruptions. To define this notion of security, we describe both a **real-world** and an **ideal-world** attack game.

The real world. In the real-world attack game, the challenger runs the setup stage just as in Attack Game 3, so that $\boldsymbol{L}$ is the set of corrupt parties and *Map* is an initially empty associative array. There are also *encryption*, *decryption*, and *corruption* queries.

- Every encryption query consists of a pair $(m, ad) \in \boldsymbol{M} \times \boldsymbol{AD}$. The challenger does the following:
  - compute $ctxt \xleftarrow{\$} E(pk, m, ad)$,
  - if $(ctxt, ad) \notin \mathsf{Domain}(Map)$, then initialize $Map[ctxt, ad] \leftarrow \emptyset$,
  - send $ctxt$ to the adversary.
- Each decryption query consists of a tuple

$$(ctxt, ad, dc, i) \in \boldsymbol{Ctxt} \times \boldsymbol{AD} \times \boldsymbol{DC} \times [N] \setminus \boldsymbol{L}.$$

  The challenger does the following:
  - if $(ctxt, ad) \in \mathsf{Domain}(Map)$, then update

$$Map[ctxt, ad] \leftarrow Map[ctxt, ad] \cup \{(dc, i)\},$$

  - compute $D(sk_i, ctxt, ad, dc)$ and send this to the adversary.
- Each corruption query specifies an index $\ell \in [N] \setminus \boldsymbol{L}$. Provided $|\boldsymbol{L}| < f$, the challenger adds ℓ to $\boldsymbol{L}$ and sends sk_ℓ to the adversary.

The ideal world. The ideal-world attack game is defined relative to a simulator *Sim*. The setup phase is the same as in the real world, except that instead of invoking $G(N, t, f)$, the challenger invokes *Sim* on input $(\texttt{setup}, N, t, f, \boldsymbol{L})$ to obtain $(pk, pkc, \{sk_\ell\}_{\ell \in \boldsymbol{L}})$.

An encryption query is processed the same as in the real world, except that instead of invoking $E(pk, m, ad)$, the challenger invokes *Sim* on input $(\texttt{enc}, |m|, ad)$, where $|m|$ is the length on m, to obtain $ctxt$. The ciphertext/associated-data pair $(ctxt, ad)$ returned by must be distinct from that associated with any previous encryption or decryption query.

A decryption query is processed the same as in the real world, except that instead of invoking $D(sk_i, ctxt, ad, dc)$, the challenger invokes *Sim* on input $(\texttt{dec}, i, ctxt, ad, dc, leakage)$. The value *leakage* will be discussed below.

A corruption query is processed the same as in the real world, expect that sk_ℓ is obtained by invoking *Sim* on input $(\texttt{cor}, \ell, leakage)$. The value *leakage* will be discussed below.

We now discuss the value *leakage* given to the simulator in processing decryption and corruption queries. Consider an encryption query with corresponding message m and ciphertext/associated-data pair $(ctxt, ad)$. Initially, we shall say that this encryption query is **protected**. Such a query

may later become **unprotected** when the values *Map* or $\boldsymbol{L}$ change in processing either a decryption or corruption query. Specifically, it becomes unprotected when

$$|\boldsymbol{L}| + \left|\{j \in [N] \setminus \boldsymbol{L} : (dc, j) \in \mathit{Map}[\mathit{ctxt}, \mathit{ad}]\}\right| \geq t,$$

for some *dc*. In processing either a decryption or corruption query, the value *leakage* is set to the set of all triples $(\mathit{ctxt}, \mathit{ad}, m)$ corresponding to encryption queries that have just become unprotected. Note that for a decryption query, this set has size at most 1.

Definition of security. We say a threshold decryption scheme is **simulation CCA secure** if there is an efficient simulator *Sim* such that no efficient adversary can effectively distinguish between the real-world attack game and the ideal-world attack game with *Sim*.

Random oracles. If the threshold scheme is based on a random oracle, then we augment both the real-world and ideal-world attack games as follows. In the real world, the challenger processes random oracle queries faithfully. In the ideal world, all random oracle queries are forwarded to *Sim*.

Static corruption. The above definition models adaptive corruptions. To model static corruptions, we simply disallow corruption queries. The adversary can still corrupt a set of parties in the setup stage.

Low-threshold security. The above definition models high-threshold security. To model low-threshold security, we simply change the definition of when an encryption query becomes unprotected. Specifically, an encryption query with corresponding ciphertext/associated-data pair $(\mathit{ctxt}, \mathit{ad})$ becomes unprotected whenever the adversary makes a decryption query $(\mathit{ctxt}, \mathit{ad}, dc, i)$ for some *dc* and $i \in [N] \setminus \boldsymbol{L}$.

Relation between game-based and simulation-based CCA security. It is fairly obvious that simulation-based CCA security implies game-based CCA security (this holds for all variants, low/high and static/adaptive). While the converse does not hold in general, it is easy to build a simulation-based CCA secure scheme E' from any game-based CCA scheme E — in the random oracle model. This is a standard type of "hybrid" construction. The scheme E' encrypts a message m with associated data ad by choosing a random key k, computing $c \leftarrow F(k) \oplus m$, and then encrypting k using the encryption algorithm for E with associated data (ad, c). Here, F is modeled as a random oracle. Moreover, it is straightforward to prove that the scheme $\mathsf{E}_{\mathrm{htdh1}}$ in Section 3 is itself simulation-based CCA secure under the LOMDH assumption, provided the underlying symmetric encryption scheme E_{s} encrypts m as $c \leftarrow F(k) \oplus m$, where (again) F is modeled as a random oracle.

Relation to universal composability. While our simulation-based definition is not explicitly given in the universal composability framework [22], it is compatible with it, and can easily be used to analyze complex systems within that framework. The essential point is the that adversary in our definition plays the role of the environment in the UC framework, while our simulator plays the role of the ideal world adversary in the UC framework.

Applications. Simulation CCA security better models a system where all ciphertexts may eventually be decrypted, but we want to ensure that information about their contents leaks only under the right conditions. For example, we already discussed the application of high-threshold context-dependent decryption to encrypted atomic broadcast in Section 7.1. Simulation-CCA security is a more natural notion of security to use in this application. Indeed, using it, one could implement an ideal functionality for atomic broadcast that works (roughly speaking) as follow:

- An honest party may submit a private transaction tx to the functionality.
- Instead of encrypting tx, the ideal functionality generates a "simulated ciphertext" $ctxt$, which does not depend on tx (other than perhaps its length). The value simply $ctxt$ plays the role of an "opaque handle" for tx that is given to the adversary.
- At some point in time, the adversary may choose to specify to the functionality that the transaction with the handle $ctxt$ should be appended to the common output sequence of transactions. Only at this point in time does the functionality leak tx to the adversary. Note that this may occur before any honest party outputs this transaction, but its location in the common output sequence is nevertheless fixed at this time.

9 Conclusions and future work

Our work introduces new security notions for CCA secure threshold decryption, as well as a new decryption context capability. This opens up new directions for future work, specifically, supporting these new capabilities in recently-introduced variants of threshold decryption. We give three examples.

Batch decryption. Choudhuri et al. [28] recently introduced the notion of *batched threshold decryption*. Their work considers the setting in which multiple ciphertexts need to be decrypted, as is the case in the applications discussed in Section 7. At a high level, they propose a mechanism by which each decrypting party can compute a single decryption share for a subset of the ciphertexts, whose size does not grow with the number of decrypted ciphertexts. If enough decryption shares are gathered for the same subset of ciphertexts, they can be decrypted. Ciphertexts outside this subset remain undecrypted and the underlying plaintexts remain hidden. This is similar in spirit to our notion of context-dependence, but with a few differences that make the notions incomparable. First, the only contexts considered by Choudhuri et al. is the subset of ciphertexts to be decrypted, whereas we support arbitrary contexts. Second, in their work, encryptions are done with respect to an epoch, and only one batch of ciphertexts can be decrypted in each epoch. On the other hand, in order to decrypt B ciphertexts in our notion of context-dependent high-threshold decryption, each decrypting party must compute B separate decryption shares and communicate all of them to the combiner, making our constructions less communication efficient.

It is thus an interesting open question whether one can get the best of both worlds: a context-dependent high-threshold decryption, in which it is possible to produce short decryption shares for batches of ciphertexts.

Silent setup. Garg et al. [41] constructed a threshold decryption scheme with *"silent setup"*, in which key generation is done locally by the decrypting parties. Concretely, each decrypting party

generates their own pair of secret and public keys. Later, the public keys of the "active" parties can be publicly aggregated to yield short public encryption and aggregation keys. Garg et al. only consider semantic security, and it is an interesting direction for future work to construct a high-threshold decryption with silent setup, both in the context-free and context-dependent settings.

More general access structures. All the definitions and constructions in this paper generalize to support more general access structures beyond threshold structures. The general techniques developed for implementing an access structure as a monotone span program [16, Ch. 22.5] should apply here directly.

Acknowledgments

This work on partially supported by NSF, the Simons Foundation, and a grant from Protocol Labs.

References

[1] Abe, M., Fehr, S.: Adaptively secure feldman vss and applications to universally-composable threshold cryptography. In: Franklin, M. (ed.) CRYPTO 2004. pp. 317–334. Springer (2004)

[2] Agarwal, A., Fernando, R., Pinkas, B.: Efficiently-thresholdizable batched identity based encryption, with applications. Cryptology ePrint Archive, Paper 2024/1575 (2024), `https://eprint.iacr.org/2024/1575`

[3] Bebel, J., Ojha, D.: Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Paper 2022/898 (2022), `https://eprint.iacr.org/2022/898`

[4] Bellare, M., Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Better than advertised security for non-interactive threshold signatures. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13510, pp. 517–550. Springer (2022). https://doi.org/10.1007/978-3-031-15985-5_18

[5] Bellare, M., Tessaro, S., Zhu, C.: Stronger security for non-interactive threshold signatures: BLS and FROST. Cryptology ePrint Archive, Paper 2022/833 (2022), `https://eprint.iacr.org/2022/833`

[6] Blakley, G.R., Meadows, C.: Security of ramp schemes. In: CRYPTO '84. Lecture Notes in Computer Science, vol. 196, pp. 242–268. Springer (1984)

[7] Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-Group signature scheme. In: PKC 2003. pp. 31–46. Springer (2002)

[8] Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Proceedings of the 2006 The Cryptographers' Track at the RSA Conference on Topics in Cryptology. p. 226–243. CT-RSA'06, Springer-Verlag (2006)

[9] Boneh, D., Bünz, B., Nayak, K., Rotem, L., Shoup, V.: Context-dependent threshold decryption and its applications. Cryptology ePrint Archive, Paper 2025/279 (2025), `https://eprint.iacr.org/2025/279`

[10] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Proc. of CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 213–229. Springer (2001). https://doi.org/10.1007/3-540-44647-8_13

[11] Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R., Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: CRYPTO 2018. p. 565–596. Springer (2018). https://doi.org/10.1007/978-3-319-96884-1_19

[12] Boneh, D., Laufer, E., Tas, E.N.: Batch decryption without epochs and its application to encrypted mempools. Cryptology ePrint Archive, Paper 2025/1254 (2025), https://eprint.iacr.org/2025/1254

[13] Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. pp. 410–428. Springer (2013)

[14] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer (2001). https://doi.org/10.1007/3-540-45682-1_30

[15] Boneh, D., Partap, A., Rotem, L.: Accountability for misbehavior in threshold decryption via threshold traitor tracing. In: CRYPTO 2024. LNCS, vol. 14926, pp. 317–351. Springer (2024). https://doi.org/10.1007/978-3-031-68394-7_11

[16] Boneh, D., Shoup, V.: A Graduate Course in Applied Cryptography (version 0.6) (January 2023), available at https://toc.cryptobook.us/

[17] Bormet, J., Faust, S., Othman, H., Qu, Z.: BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. ePrint 2024/1533 (2024)

[18] Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS '05. p. 320–329 (2005). https://doi.org/10.1145/1102120.1102162

[19] Buchman, E., Kwon, J., Milosevic, Z.: The latest gossip on BFT consensus (2018), arXiv:1807.04938, http://arxiv.org/abs/1807.04938

[20] Cachin, C., Kursawe, K., Shoup, V.: Random oracles in Constantinople: practical asynchronous byzantine agreement using cryptography (extended abstract). In: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing. pp. 123–132 (2000)

[21] Camenisch, J., Drijvers, M., Hanke, T., Pignolet, Y., Shoup, V., Williams, D.: Internet computer consensus. In: PODC '22. pp. 81–91. ACM (2022). https://doi.org/10.1145/3519270.3538430

[22] Canetti, R.: Universally composable security. J. ACM **67**(5), 1–94 (2020)

[23] Canetti, R., Goldwasser, S.: An efficient *Threshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In: Stern, J. (ed.) EUROCRYPT '99. Lecture Notes in Computer Science, vol. 1592, pp. 90–106. Springer (1999). https://doi.org/10.1007/3-540-48910-X_7

[24] Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: EUROCRYPT 2004. pp. 207–222. Springer (2004)

[25] Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002). https://doi.org/10.1145/571637.571640

[26] Chan, B.Y., Pass, R.: Simplex consensus: A simple and fast consensus protocol. In: TCC 2023. Lecture Notes in Computer Science, vol. 14372, pp. 452–479. Springer (2023). https://doi.org/10.1007/978-3-031-48624-1_17

[27] Chen, H., Cramer, R., Goldwasser, S., de Haan, R., Vaikuntanathan, V.: Secure computation from random error correcting codes. In: EUROCRYPT 2007. Lecture Notes in Computer Science, vol. 4515, pp. 291–310. Springer (2007)

[28] Choudhuri, A.R., Garg, S., Piet, J., Policharla, G.: Mempool privacy via batched threshold encryption: Attacks and defenses. In: USENIX Security 2024 (2024)

[29] Choudhuri, A.R., Garg, S., Policharla, G.V., Wang, M.: Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive, Paper 2024/1516 (2024), `https://eprint.iacr.org/2024/1516`

[30] Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Proceedings of the Second International Conference on Theory of Cryptography. p. 342–362. TCC'05, Springer-Verlag (2005). https://doi.org/10.1007/978-3-540-30576-7_19

[31] Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: CRYPTO 1998. p. 13–25. Springer-Verlag (1998)

[32] Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: IEEE Computer Society Press. pp. 910–927 (2020)

[33] Das, S., Ren, L., Yang, Z.: Adaptively secure threshold ElGamal decryption from DDH. Cryptology ePrint Archive, Paper 2025/1477 (2025), `https://eprint.iacr.org/2025/1477`

[34] De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing. p. 522–533. STOC '94, Association for Computing Machinery (1994). https://doi.org/10.1145/195058.195405

[35] Desmedt, Y.: Society and group oriented cryptography: A new concept. In: CRYPTO 1987. p. 120–127. Springer-Verlag (1987)

[36] Desmedt, Y.: Threshold cryptosystems. In: AUSCRYPT '92. pp. 1–14. Springer Berlin Heidelberg (1992)

[37] Devevey, J., Libert, B., Nguyen, K., Peters, T., Yung, M.: Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In: PKC 2021. LNCS, vol. 12710, pp. 659–690. Springer (2021). https://doi.org/10.1007/978-3-030-75245-3_24

[38] Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: TCC'05. p. 188–209. Springer (2005). https://doi.org/10.1007/978-3-540-30576-7_11

[39] Dziembowski, S., Faust, S., Luhn, J.: Shutter network: Private transactions from threshold cryptography. ePrint 2024/1981 (2024), https://eprint.iacr.org/2024/1981

[40] Fouque, P.A., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: ASIACRYPT 2001. p. 351–368. Springer-Verlag (2001)

[41] Garg, S., Kolonelos, D., Policharla, G., Wang, M.: Threshold encryption with silent setup. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024. Lecture Notes in Computer Science, vol. 14926, pp. 352–386. Springer (2024)

[42] Groth, J.: Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Paper 2021/339 (2021), https://eprint.iacr.org/2021/339

[43] Jalalzai, M.M., Niu, J., Feng, C.: Fast-hotstuff: A fast and resilient hotstuff protocol (2020), arXiv:2010.11454, http://arxiv.org/abs/2010.11454

[44] Jarecki, S., Lysyanskaya, A.: Adaptively secure threshold cryptography: introducing concurrency, removing erasures. In: EUROCRYPT'00. p. 221–242. Springer (2000)

[45] Labs, O.: Osmosis project (2021), https://docs.osmosis.zone/

[46] Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and cca2-secure encryption from homomorphic signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. pp. 514–532. Springer (2014)

[47] Libert, B., Yung, M.: Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In: Theory of Cryptography 2012. LNCS, vol. 7194, pp. 75–93. Springer (2012). https://doi.org/10.1007/978-3-642-28914-9_5

[48] Libert, B., Yung, M.: Adaptively secure non-interactive threshold cryptosystems. Theoretical Computer Science **478**, 76–100 (2013). https://doi.org/https://doi.org/10.1016/j.tcs.2013.01.001

[49] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT 1999. p. 223–238. Springer-Verlag (1999)

[50] Reiter, M., Birman, K.: How to securely replicate services. ACM TOPLAS **16**(3), 986–1009 (1994)

[51] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Proc. of CRYPTO '84. Lecture Notes in Computer Science, vol. 196, pp. 47–53. Springer (1984). https://doi.org/10.1007/3-540-39568-7_5

[52] Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer (2000). https://doi.org/10.1007/3-540-45539-6_15

[53] Shoup, V.: Back to the future: simple threshold decryption secure against adaptive corruptions. Cryptology ePrint Archive, Paper 2025/1578 (2025), https://eprint.iacr.org/2025/1578

[54] Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptol. **15**(2), 75–96 (2002)

[55] Suegami, S., Ashizawa, S., Shibano, K.: Constant-cost batched partial decryption in threshold encryption. ePrint 2024/762 (2024), `https://eprint.iacr.org/2024/762`

[56] de Valence, H.: The Penumbra protocol (2021)

[57] Wee, H.: Threshold and revocation cryptosystems via extractable hash proofs. In: EUROCRYPT'11. p. 589–609. Springer-Verlag (2011)

A A Threshold IBE Scheme

In this section we present a threshold variant of the Boneh-Franklin identity-based encryption (IBE) scheme [10], satisfying the conditions needed for the construction in Section 6. The scheme $\mathsf{TBF} = (G, K, E, D)$ is parameterized in terms of an identity space $\boldsymbol{ID}$, and a message space $\boldsymbol{M}$ which we assume to be an abelian group with a group operation $\circ$ (for example $\boldsymbol{M} = \{0,1\}^n$ for some n with $\circ$ being the xor operation $\oplus$). The construction makes use of the following building blocks:

- a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathcal{G}_1, \mathcal{G}_2, e)$ of prime order q, where $\mathbb{G}_1$ is generated by $\mathcal{G}_1 \in \mathbb{G}_1$, $\mathbb{G}_2$ is generated by $\mathcal{G}_2 \in \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ is a non-degenerate bilinear map;
- two hash functions (which will be modeled as random oracles):
  - for *identity group derivation*, $H_{\mathrm{id}} : \boldsymbol{ID} \to \mathbb{G}_1$,
  - for *encryption key derivation*, $H_{\mathrm{kd}} : \mathbb{G}_T \to \boldsymbol{M}$.

The threshold IBE scheme TBF is essentially the standard Boneh-Franklin IBE scheme, where the master secret key is shared among the parties using a t-out-of-N Shamir secret sharing. It is defined as follows.

- Algorithm $G(N, t)$ computes the master public key and master secret keys:

$1:$ $x \leftarrow\!\$\ \mathbb{Z}_q$ // sample a master secret key
$2:$ $a_1, \ldots, a_{t-1} \leftarrow\!\$\ \mathbb{Z}_q$ // sample polynomial coefficients for Shamir secret sharing
$3:$ **for** $i \in [N]$:
$4:$ $\quad msk_i \leftarrow x + \sum_{j=1}^{t} a_j \cdot i^j$ // msk_i is the ith secret share of x
$5:$ $\quad pk_i \leftarrow msk_i \mathcal{G}_2$ // pk_i is the ith public key, used for robustness
$6:$ $\mathcal{X} \leftarrow x\mathcal{G}_2$ // $\mathcal{X}$ will be the master public key is in $\mathbb{G}_2$
$7:$ $mpk \leftarrow \mathcal{X}$
$8:$ $pkc \leftarrow (pk_1, \ldots, pk_N)$
$9:$ **output** $(mpk, pkc, msk_1, \ldots, msk_N)$

- Algorithm $K(msk_i, id)$ uses msk_i to derive a key share for id:

  1 : $\mathcal{U}_{id} \leftarrow H_{\mathrm{id}}(id)$ // hash id to $\mathbb{G}_1$
  2 : $\mathcal{S}_{id,i} \leftarrow msk_i\, \mathcal{U}_{id}$ // compute the ith component of the identity key for id in $\mathbb{G}_1$
  3 : **output** $sk_{id,i} \coloneqq \mathcal{S}_{id,i}$

- Algorithm $E(mpk, id, m)$ uses $mpk = \mathcal{X}$ to encrypt m to identity id:

  1 : $\mathcal{U}_{id} \leftarrow H_{\mathrm{id}}(id)$ // hash id to $\mathbb{G}_1$
  2 : $r \leftarrow\!\!\$\, \mathbb{Z}_p$ // sample randomness for encryption
  3 : $\mathcal{R} \leftarrow r\mathcal{G}_2$
  4 : $\mathcal{Y} \leftarrow e(\mathcal{U}_{id}, r\mathcal{X})$ // compute a group element in $\mathbb{G}_T$ to derive a one-time pad key
  5 : $k \leftarrow H_{\mathrm{kd}}(\mathcal{Y})$ // derive a one-time pad key
  6 : $c \leftarrow k \circ m$ // $\circ$ denotes the group operation in $\boldsymbol{M}$
  7 : **output** $ctxt \coloneqq (c, \mathcal{R}) \in \boldsymbol{M} \times \mathbb{G}_2$

- Algorithm $D(pkc, \boldsymbol{J}, \{sk_{id,j}\}_{j\in\boldsymbol{J}}, ctxt)$ decrypts a ciphertext $ctxt = (c, \mathcal{R})$ with a combiner key $pkc = (pk_1, \ldots, pk_N)$ and identity key components $sk_{id,j} = \mathcal{S}_{id,j}$:

  1 : $\boldsymbol{J}^* \leftarrow \emptyset$ // initialize a set of malformed key shares
  2 : **for** $j \in \boldsymbol{J}$:
  3 : **if** $e(H_{\mathrm{id}}(id), pk_j) \neq e(sk_{id,j}, \mathcal{G}_2)$
  4 : **then** $\boldsymbol{J}^* \leftarrow \boldsymbol{J} \cup \{j\}$ // if the pairing check fails, add j to the set of malformed shares
  5 : **if** $\boldsymbol{J}^* \neq \emptyset$
  6 : **then output** $\texttt{blame}(\boldsymbol{J}^*)$
  7 : $\mathcal{S}_{id} \leftarrow \sum_{j\in\boldsymbol{J}} \lambda_j^{(\boldsymbol{J})} \mathcal{S}_{id,j}$ // interpolate the id's identity key with Lagrange coefficients $\{\lambda_j^{(\boldsymbol{J})}\}_{j\in\boldsymbol{J}}$
  8 : $\mathcal{Y} \leftarrow e(\mathcal{S}_{id}, \mathcal{R})$ // compute a group element in $\mathbb{G}_T$ to derive a one-time pad key
  9 : $k \leftarrow H_{\mathrm{kd}}(\mathcal{Y})$ // derive a one-time pad key
  10 : $m \leftarrow c \circ k^{-1}$ // undo the one-time pad in $\boldsymbol{M}$
  11 : **output** m

Robustness. Observe that the construction satisfied both properties needed for robustness: accurate blaming (Definition 7) and consistency (Definition 8). The key derivation algorithm K is deterministic, and for every combiner key pkc, identity id, and $i \in \{1, \ldots, N\}$, it always outputs the same key share $sk_{id,i}$. The pairing check done by D always passes for this honestly-generated key share, since $e(H_{\mathrm{id}}(id), pk_i) = e(sk_{id,i}, \mathcal{G}_2) = e(H_{\mathrm{id}}(id), \mathcal{G}_2)^{msk_i}$. This establishes accurate blaming. Moreover, this pairing check fails for all other key shares $sk'_{id,i} \neq sk_{id,i}$. Hence, if two collections of key shares $\{sk_{id,j}\}_{j\in\boldsymbol{J}_1}$ and $\{sk'_{id,j}\}_{j\in\boldsymbol{J}_2}$ for an identity id and two subsets $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ result in decryption of a ciphertext $ctxt$ into messages m_1 and m_2, then by the correctness of Lagrange interpolation, it must be that $m_1 = m_2$.

Security. The semantic security of the threshold Boneh-Franklin IBE relies on a slight strengthening of the LOMDH assumption presented in Section 4, that we call **the Linear One More Bilinear Diffie Hellman (LOMBDH) Assumption**. The LOMBDH assumption is defined via a the following attack game.

In this game, a challenger first chooses $s_1, \ldots, s_k$, $t_1, \ldots, t_\ell$, and $r_1, \ldots, r_n$ at random from $\mathbb{Z}_q$, and gives to the adversary the group elements

$$\mathcal{S}_i \coloneqq s_i \mathcal{G}_1 \in \mathbb{G}_1 \quad (i = 1, \ldots, k)$$

and

$$\mathcal{T}_j \coloneqq t_j \mathcal{G}_1 \in \mathbb{G}_1 \quad (j = 1, \ldots, \ell)$$

and

$$\mathcal{R}_j \coloneqq r_j \mathcal{G}_2 \in \mathbb{G}_2 \quad (j = 1, \ldots, n).$$

Next, the adversary makes a sequence of **linear DH-queries**. Each such query consists of a matrix of scalars $\{\kappa_{i,j}\} \in \mathbb{Z}_q^{[k]\times[\ell]}$, to which the challenger responds with the group element

$$\sum_{i,j} \kappa_{i,j} s_i t_j \mathcal{G} = \sum_{i,j} \kappa_{i,j} \mathsf{DH}(\mathcal{S}_i, \mathcal{T}_j).$$

At the end of the game, the adversary outputs a list of triples, each of the form

$$(\mathcal{V}, \{\kappa_{i,j},\}, w) \in \mathbb{G}_T \times \mathbb{Z}_q^{[k]\times[\ell]} \times \{1, \ldots, n\}.$$

The adversary wins the game if for one such output group/matrix pair, we have

(i) $\{\kappa_{i,j}\}$ is not in the linear span of the input matrices, and

(ii)
$$\mathcal{V} = e(\mathcal{G}_1, \mathcal{G}_2)^{r_w \cdot \sum_{i,j} \kappa_{i,j} s_i t_j}$$

We say that the LOMBDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathcal{G}_1, \mathcal{G}_2, e)$ if no probabilistic polynomial-time adversary can win in the above game with non-negligible probability.

The following theorem establishes the semantic security of TBF per Definition 9 in the random oracle model, relying on the LOMBDH assumption.

Theorem 5. *If H_{id} and H_{kd} are modeled as random oracles, and if the LOMBDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathcal{G}_1, \mathcal{G}_2, e)$, then the threshold identity-based encryption scheme* TBF *is semantically secure.*

The proof of Theorem 5, as it is almost identical to the proof of Theorem 4.1 in the full version [5] of Bellare et al. [4].