DKG Is All You Need

Guru-Vamsi Policharla
Commonware, Inc.

Abstract

We construct the first Batched Threshold Encryption scheme with a *transparent* setup where public parameters are *independent* of the batch size. As a result batches of arbitrary sizes can be decrypted, without imposing an a priori fixed bound. We prove security under a constant size assumption – the decisional bilinear square Diffie–Hellman assumption.

Setup is just a distributed key generation protocol to sample secret shares of a random value. Ciphertexts consist of two $\mathbb{G}_1$ elements, one $\mathbb{G}_2$ element and the encrypted message, plus a NIZK for CCA security (two $\mathbb{F}$ elements with a sigma protocol). Partial decryptions are a single $\mathbb{G}_1$ element, computed with one scalar multiplication. Decrypting a batch of B ciphertexts costs $O(B)$ pairings and $O(B \log^2 B)$ group operations.

1 Introduction

Batched threshold encryption [CGPP24] (BTE) allows any t-out-of-n parties in a committee to decrypt a batch of B ciphertexts using communication that is sub-linear in B. The main application of BTE is in building encrypted mempools to protect the privacy of pending transactions in decentralized trading platforms. Given the challenging nature of building distributed systems, we ideally want the setup requirements to be minimal – using only off-the-shelf primitives that are already implemented in modern blockchains, such as distributed key generation without relying on trusted dealers.

But so far, all known BTE schemes have a) public parameters that grow with the maximum batch size and b) rely on a structured reference string or an interactive multi-round setup. In this work, we ask:

Can we realize Batched Threshold Encryption with short public parameters and just a DKG?

Our Contributions. We construct the first BTE scheme where public parameters are independent of the batch size. Additionally:

- per the title, it is the first scheme where the setup requirement is just a DKG for a random element
- we prove security under a constant size hardness assumption (Definition 6)
- we observe that our scheme can bootstrap any (weighted and/or silent) threshold encryption scheme to support batched decryption if the ciphertexts are linearly homomorphic with message space $\mathbb{F}$

Overview. We now provide an overview of our construction and discuss extensions to the weighted (and silent setup) setting. Let $[1]_1, [1]_2$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively and g' be a generator of $\mathbb{G}_1$ with an unknown discrete logarithm. The committee runs a DKG to sample a secret sharing of $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{F}$ and publish $\mathsf{pk} = [\mathsf{sk}]_1$ and $\{\mathsf{pk}_i = [\mathsf{sk}_i]_1\}_{i\in[n]}$ where sk_i denotes the i-th party's share of sk.

Encrypt. To encrypt a message $m \in \mathbb{G}_T$, sample $k \leftarrow\!\!\$\ \mathbb{F}$ and output

$$\mathsf{ct} := \left([k]_1,\ [k]_2,\ k \cdot (g' + x \cdot \mathsf{pk}),\ k^2 \cdot [\mathsf{sk}]_T + m\right)$$

where $x \leftarrow H([k]_1)$, and $H : \mathbb{G}_1 \to \mathbb{F}$ is a collision-resistant hash function (not modeled as a random oracle).

Partial Decryption. To decrypt a batch of ciphertexts

$$\{\mathsf{ct}^{(i)} = ([k_i]_1,\ [k_i]_2,\ k_i \cdot (g' + x_i \cdot \mathsf{pk}),\ k_i^2 \cdot [\mathsf{sk}]_T + m_i)\}_{i\in[B]}$$

the j-th committee member publishes

$$\mathsf{pd}_j := \mathsf{sk}_j \cdot \sum_{i=1}^{B} [k_i]_1$$

Aggregation. Given t partial decryptions, recover $\mathsf{pd} = \mathsf{sk} \cdot \sum_{i=1}^{B} [k_i]_1$ by interpolating in the exponent. To recover the i-th message, compute $[k_i^2 \cdot \mathsf{sk}]_T$ as follows:

- compute $\mathsf{pd} \circ [k_i]_2 = [k_i^2 \mathsf{sk}]_T + \sum_{j\neq i} [k_i k_j \mathsf{sk}]_T$
- for $j \neq i$, compute the cross terms as:

$$[k_i k_j \mathsf{sk}]_T = \frac{(k_j \cdot (g' + x_j \cdot \mathsf{pk})) \circ [k_i]_2 - (k_i \cdot (g' + x_i \cdot \mathsf{pk})) \circ [k_j]_2}{x_j - x_i}$$

Thus we can recover the i-th message as:

$$m_i = \mathsf{ct}_4^{(i)} - \mathsf{pd} \circ [k_i]_2 + \sum_{j\neq i} \frac{(k_j \cdot (g' + x_j \cdot \mathsf{pk})) \circ [k_i]_2 - (k_i \cdot (g' + x_i \cdot \mathsf{pk})) \circ [k_j]_2}{x_j - x_i}$$

Implemented naively, the aggregation step requires $O(B^2)$ group operations but in Section 4 we discuss how this can be reduced to $O(B \log^2 B)$ group operations and $O(B)$ pairings using fast polynomial arithmetic. As in prior work, we can attach to each ciphertext a simulation-extractable NIZK proof of knowledge of k to obtain CCA security, which can be instantiated with a Sigma protocol.

Bootstrapping. We now describe how our construction can bootstrap *any* threshold encryption scheme with linearly homomorphic ciphertexts and message space $\mathbb{F}$ to a batched threshold encryption scheme. In fact, BEAST-MEV [BCF+25] uses a similar insight in the context of [BFOQ25] but the resulting scheme had position-dependent ciphertexts.

Observe that in the scheme above, once a ciphertext has been chosen to be decrypted (by t parties), the randomness k chosen by the encryptor no longer serves any purpose and can actually be revealed. Furthermore, revealing $\sum_{i\in[B]} k_i$ is sufficient to decrypt the entire batch of ciphertexts: anyone can compute the partial decryption $\mathsf{pd} = (\sum_i k_i)[\mathsf{sk}]_1$, and $[\mathsf{sk}]_1$ can be set to a random group element with an unknown discrete logarithm.

To extend a (weighted and/or silent) linearly homomorphic threshold encryption scheme Π to support batched decryption, encrypt the randomness k under Π's public key and aggregate the resulting ciphertexts into an encryption of $\sum_{i\in[B]} k_i$. Each committee member releases one partial decryption to recover the aggregated randomness, which enables decryption of all B messages. Importantly, such a composition does not introduce epoch restrictions or position-dependent ciphertexts as seen in earlier BTE schemes. We refer to BEAST-MEV [BCF+25, Sections 6–7] for details on CCA security. When combined with [Pol26c] it yields a weighted batched threshold encryption scheme with silent setup with public parameters $O(nW)$ and $O(\lambda/\log\lambda)$ ciphertext size, where W is the maximum supported total committee weight.[1]

[1]The larger ciphertexts are needed for $\mathbb{F}$-plaintext recovery by solving the bounded discrete logarithm problem. Partial decryptions also grow with the number of chunks, while remaining independent of the batch size.

[2]The construction of [Pol26c] can be extended to the batched setting with the listed parameters (personal communication with the authors).

Construction	Setup	|pp|	|ct|	E	CR	Assumption
BTE [CGPP24]	MPC (per epoch)	$O(B+n)$	$2[\mathbb{G}_1]+[\mathbb{G}_2]$	✗	✗	q-SBDHT
BIBE [CGPW25], [AFP25]	SRS + DKG	$O(B+n)$	$3[\mathbb{G}_1]$	✗	✓	GGM
[SAS24]	SRS + DKG	$O(B+n)$	$2[\mathbb{G}_1]$	✗	✗	B-dlog + AGM
BEAT-MEV [BFOQ25]	SRS + DKG	$O(B^2+n)$	$3[\mathbb{G}_1]$	✓	✗	DBPDH
	SRS + DKG	$O(B+n)$	$3[\mathbb{G}_1]$	✓	✗	B-DBPDH
TrX [FPTX25]	SRS + DKG	$O(kB+n)$	$2[\mathbb{G}_1]$	✓	✓	GGM
[GWWW25]	SRS + DKG	$O(B+n)$	$2[\mathbb{G}_1]$	✗	✓	GGM
Multi-Key BIBE [Pol26a]	SRS + DKG	$O(qB+n)$	$3[\mathbb{G}_1]$	q	✓	GGM
BEAT-MEV++ [ABD+25]	SRS + DKG	$O(Bn)$	$2[\mathbb{G}_1]$	✓	✗	B-DBPDH + AGM
[GWWW25]	SRS + Silent	$O(Bn^2)$	$2[\mathbb{G}_1]+2[\mathbb{G}_2]$	✗	✓	q-type
BEAST-MEV [BCF+25]	SRS + Silent	$O(B+n^2)$	$O(\lambda/\log\lambda)$	✓	✗	GGM
[ACADG26]	SRS + Silent	$O(n^2+Bn)$	$2[\mathbb{G}_1]$	✓	✓	GGM
[Pol26c][2]	SRS + Silent	$O(n^2+Bn)$	$2[\mathbb{G}_1]$	✓	✓	GGM
Partial-Fraction BTE [BNRT26]	MPC	$O(Bn)$	$2[\mathbb{G}_1]$	✓	✓	B-StatRankDef
	MPC	$O(Bn)$	$[\mathbb{G}_1]$	✓	✓	B-QuadRankDef
Simple BTE [Pol26b], BTX [ADG+26]	MPC	$O(Bn)$	$[\mathbb{G}_1]$	✓	✓	B-DBPDH
[FPRT26]	Two Round	$O(Bn)$	$[\mathbb{G}_1]$	✓	✓	B-DBPDH
This work	DKG	$O(n)$	$2[\mathbb{G}_1]+[\mathbb{G}_2]$	✓	✓	DBSDH
This work + [Pol26c]	SRS + Silent	$O(n^2)$	$O(\lambda/\log\lambda)$	✓	✓	GGM

Table 1: Comparison of pairing-based batched threshold encryption schemes in the unweighted setting. See the text for the notation.

Comparison with prior work. Table 1 compares pairing-based batched threshold encryption schemes in the unweighted setting. [3] The column |pp| is the total size of the public parameters, including any structured reference string, all public keys, and public aggregation hints. The column |ct| excludes the encrypted message (a $\mathbb{G}_T$ element or a symmetric ciphertext) and NIZK proofs for CCA security. Where the construction permits, we exchange the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ to minimize ciphertext size which may change decryption-share sizes and operation costs.

The *Setup* column distinguishes five kinds of setup. "SRS + DKG" requires a structured reference string (e.g., powers of tau) in addition to a DKG for the shared secret. "SRS + Silent" requires an SRS but no interaction between the parties. "MPC" requires an interactive setup with secure multiplications. "Two Round" is the specialized two-round DKG of [FPRT26], whose output has size $O(Bn)$ and must be regenerated when the maximum batch size changes. Finally, "DKG" refers to a Distributed Key Generation protocol to sample a Shamir secret sharing of a random value.

The two BEAT-MEV rows correspond to the quadratic-size puncturable-PRF setup under a constant-size assumption and the linear-size variant under B-DBPDH, respectively. The column E indicates that the scheme does not suffer from epoch restrictions, where q means that a ciphertext can be decrypted in any of q consecutive batches, and CR indicates censorship resistance. Throughout, B is the maximum batch size (our scheme has no such bound), n the committee size and k an a priori bound on the number of batches that can be decrypted. Schemes that support weighted thresholds [ABD+25, FPRT26, ACADG26, Pol26c], including our silent extension, are listed with all weights equal to one.

[3] See also lattice-based constructions [BLT25, XZC26].

2 Preliminaries

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. The security parameter is denoted by $\lambda \in \mathbb{N}$. A function $f : \mathbb{N} \to \mathbb{N}$ is polynomially bounded if $f(n) = O(n^c)$ for some constant $c \geq 0$ as $n \to \infty$, and we write $\mathsf{poly}(\cdot)$ to denote a polynomial bound. A function $f : \mathbb{N} \to [0, 1]$ is said to be negligible if for every $c \in \mathbb{N}$, there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$, and we write $\mathsf{negl}(\cdot)$ to denote such a function. A probability is *noticeable* if it is not negligible, and *overwhelming* if it is equal to $1 - \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\lambda)$. For a set $\mathcal{S}$, we write $s \leftarrow\!\$\ \mathcal{S}$ to indicate that s is sampled uniformly at random from $\mathcal{S}$. An algorithm $\mathcal{A}$ is PPT (probabilistic polynomial-time) if its running time is bounded by some polynomial in the size of its input.

Bilinear groups. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \circ) \leftarrow \mathsf{GrpGen}(1^\lambda)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p written additively, $[1]_1, [1]_2$ are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, and $\circ : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable bilinear map satisfying (i) *bilinearity*: $[x]_1 \circ [y]_2 = [xy]_T$ for all $x, y \in \mathbb{F}$, and (ii) *non-degeneracy*: $[1]_1 \circ [1]_2 \neq 0_{\mathbb{G}_T}$. Here $\mathbb{F} := \mathbb{F}_p$ denotes the scalar field, $[1]_T := [1]_1 \circ [1]_2$, and for $b \in \{1, 2, T\}$ we write $[x]_b := x \cdot [1]_b$. We work in the Type-III (asymmetric) setting: no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is assumed, although our construction does not rely on its absence.

Non-interactive Zero-Knowledge proofs. Let $\mathcal{L}$ be an NP-language and $\mathcal{R}$ the corresponding NP-relation. A simulation-extractable NIZK (SE-NIZK) for $\mathcal{R}$ consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{crs}, \mathsf{td})$: Takes as input a security parameter, and outputs a common reference string crs and trapdoor td.
- $\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$: Takes as input crs and any pair $(x, w) \in \mathcal{R}$ and outputs a proof π for the statement $x \in \mathcal{L}$.
- $\mathsf{Verify}(\mathsf{crs}, x, \pi) \to b$: Takes as input crs, statement x and proof π and outputs a bit b indicating whether verification has passed or failed.
- $\mathsf{SimProve}(\mathsf{crs}, \mathsf{td}, x) \to \pi$: Takes as input crs, trapdoor td and statement x and outputs a *simulated* proof π.

We will drop the crs term wherever implicit. The NIZKs we use have a transparent setup, so the crs only consists of public parameters. When specifying a relation, we write $\{w \mid \varphi_1 \wedge \cdots \wedge \varphi_m\}$ where w is the witness and each φ_i is a constraint. At times we will have trivial constraints which just contain an unconstrained variable. In this case, the unconstrained variable is simply part of the statement, so that the proof is bound to it. A SE-NIZK satisfies the following properties:

- *Perfect Completeness.* An SE-NIZK satisfies perfect completeness if for any $(x, w) \in \mathcal{R}$, we have

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{array} : \mathsf{Verify}(x, \pi) = 1\right] = 1.$$

- *Proof of knowledge (and soundness).* For every PPT adversary $\mathcal{A}$, there is a PPT extractor Ext such that

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w \leftarrow \mathsf{Ext}(\mathsf{crs}, x, \pi)\end{array} : \begin{array}{c}\mathsf{Verify}(x, \pi) = 1 \\ (x, w) \notin \mathcal{R}\end{array}\right] \leq \mathsf{negl}(\lambda).$$

- *Computational Zero-knowledge.* There exists an algorithm $\mathsf{SimProve}$ such that for all PPT adversaries $\mathcal{A}$:

$$\left|\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{array} : \begin{array}{c}(x, w) \in \mathcal{R} \\ \mathcal{A}(\pi) = 1\end{array}\right] - \Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}) \\ \pi \leftarrow \mathsf{SimProve}(\mathsf{crs}, \mathsf{td}, x)\end{array} : \begin{array}{c}(x, w) \in \mathcal{R} \\ \mathcal{A}(\pi) = 1\end{array}\right]\right| \leq \mathsf{negl}(\lambda).$$

- *Simulation-Extractability.* For every PPT adversary $\mathcal{A}$, there exists a PPT extractor Ext such that

$$\Pr\left[\begin{array}{c|c} & \mathsf{Verify}(x,\pi)=1 \\ (x,\pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}(\mathsf{crs},\mathsf{td},\cdot)}(\mathsf{crs}) : & \wedge (x,w) \notin \mathcal{R} \\ & \wedge (x,\pi) \notin Q \end{array}\right] \leq \mathsf{negl}(\lambda).$$

  where $\mathcal{A}$ has oracle access to $\mathsf{SimProve}(\mathsf{crs},\mathsf{td},\cdot)$, and Q is the list of statement-proof pairs returned by that oracle.

We will also demand that the proof is straight-line extractable. This means that the extractor can, on input a valid proof and (potentially) the list of random-oracle queries made by the adversary (prover), extract a witness with overwhelming probability without rewinding the prover. When the SE-NIZK is instantiated in the random-oracle model, the crs may be taken as empty and extraction uses the adversary's random-oracle queries.

3 Definitions

We recall the definitions of batched threshold encryption. Throughout, the committee size $n = n(\lambda)$, threshold $t = t(\lambda)$, and batch lengths are polynomially bounded in λ, with $1 \leq t \leq n$.

Definition 1 (Batched Threshold Encryption)**.** *A* batched threshold encryption (BTE) *scheme instantiated with $n \in \mathbb{N}$ parties, a threshold of $t \in [n]$, and message space of $\mathcal{M}$ consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda, 1^n, 1^t) \to (\mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{dk})$*: The setup algorithm takes the security parameter λ, the number of parties n, and the threshold t, and outputs a public encryption key* ek*, a public decryption key* dk*, and n secret keys* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.
- $\mathsf{Enc}(\mathsf{ek}, m) \to \mathsf{ct}$*: The encryption algorithm takes the encryption key* ek *and a message $m \in \mathcal{M}$, and outputs a ciphertext* ct.
- $\mathsf{PartialDec}(\mathsf{ek}, \mathsf{sk}_j, \{\mathsf{ct}_i\}_{i\in[B]}) \to \mathsf{pd}_j$*: The (deterministic) partial-decryption algorithm takes the public encryption key* ek*, a secret key* sk_j *for a party $j \in [n]$, and B ciphertexts, and outputs a partial decryption share* pd_j *or $\perp$.*
- $\mathsf{Verify}(\mathsf{dk}, j, \mathsf{pd}_j, \{\mathsf{ct}_i\}_{i\in[B]}) \to \{0,1\}$*: The (deterministic) verification algorithm takes the decryption key* dk*, a party index $j \in [n]$, a claimed partial decryption share* pd_j*, and B ciphertexts, and outputs 1 iff* pd_j *is a valid share for party j on that batch.*
- $\mathsf{Combine}(\mathsf{dk}, T \subseteq [n], \{\mathsf{pd}_j\}_{j\in T}) \to \mathsf{pd}$*: The (deterministic) combiner algorithm takes the decryption key* dk*, a subset $T \subseteq [n]$, and the partial decryption shares of these $|T|$ parties, and outputs a partial decryption* pd *or $\perp$.*
- $\mathsf{Dec}(\mathsf{dk}, \mathsf{pd}, \{\mathsf{ct}_i\}_{i\in[B]}) \to \{m_i\}_{i\in[B]}$*: The (deterministic) decryption algorithm takes the decryption key* dk*, a partial decryption* pd *and B ciphertexts, and outputs the corresponding messages, where each $m_i \in \mathcal{M} \cup \{\perp\}$.*

Let $\mathsf{BTE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{PartialDec}, \mathsf{Verify}, \mathsf{Combine}, \mathsf{Dec})$ be a BTE scheme.

Definition 2 (Correctness of BTE)**.** *We say* BTE *is* correct *if for all polynomially bounded $B = B(\lambda) \geq 1$, $n = n(\lambda)$, and $t = t(\lambda)$ with $1 \leq t \leq n$, all $T \subseteq [n]$ such that $|T| \geq t$, and $\{m_i\}_{i\in[B]} \in \mathcal{M}^B$,*

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(\mathsf{dk}, j, \mathsf{pd}_j, \{\mathsf{ct}_i\}_{i\in[B]}) = 1 \ \forall j \in T & (\mathsf{ek}, \{\mathsf{sk}_j\}_{j\in[n]}, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^t) \\ \quad \wedge \mathsf{Dec}(\mathsf{dk}, \mathsf{pd}, \{\mathsf{ct}_i\}_{i\in[B]}) = \{m_i\}_{i\in[B]} & \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}, m_i) \ \forall i \in [B] \\ & \mathsf{pd}_j \leftarrow \mathsf{PartialDec}(\mathsf{ek}, \mathsf{sk}_j, \{\mathsf{ct}_i\}_{i\in[B]}) \ \forall j \in T \\ & \mathsf{pd} \leftarrow \mathsf{Combine}(\mathsf{dk}, T, \{\mathsf{pd}_j\}_{j\in T}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 3 (B-IND-CCA Security of BTE)**.** BTE *is* B-IND-CCA-secure *if all non-uniform PPT adversaries $\mathcal{A}$ have negligible advantage in the following game:*

- *$\mathcal{A}$ first outputs a subset $S \subseteq [n]$ of size $t-1$ that it wishes to corrupt.*
- *The challenger runs $(\mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^t)$ and sends ek, dk, and $\{\mathsf{sk}_j\}_{j \in S}$ to $\mathcal{A}$.*
- ***Pre-challenge queries:** The adversary may adaptively issue a polynomial number of queries for computing partial decryption shares. For each query, $\mathcal{A}$ chooses a batch length $B \geq 1$ and sends a list $\{\mathsf{ct}_i\}_{i \in [B]}$ of B ciphertexts. The challenger computes $\mathsf{pd}_j := \mathsf{PartialDec}(\mathsf{ek}, \mathsf{sk}_j, \{\mathsf{ct}_i\}_{i \in [B]})$ for $j \in [n] \setminus S$ and returns the outputs to $\mathcal{A}$.*
- ***Challenge round:** $\mathcal{A}$ sends two messages $m_0, m_1 \in \mathcal{M}$. The challenger samples $b \leftarrow\$ \{0,1\}$, computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$, and sends ct^* to $\mathcal{A}$.*
- ***Post-challenge queries:** The adversary may continue to issue queries as above, choosing the batch length for each query. If $\mathcal{A}$ sends a list $\{\mathsf{ct}_i\}_{i \in [B]}$ such that $\mathsf{ct}^* \in \{\mathsf{ct}_i\}_{i \in [B]}$, the challenger returns $\perp$. Otherwise it answers as in the pre-challenge phase.*
- ***Output:** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

The adversary's advantage is

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BTE},\mathcal{A}}(\lambda, n, t) := \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

Robustness. A BTE scheme must be robust against malicious decryption servers or clients that submit malformed ciphertexts or decryption queries. Informally, we require the following two properties:

- **Consistency:** It should be computationally infeasible for an adversary to produce two different sets of partial decryptions for the same batch of ciphertexts such that both sets verify yet Combine yields conflicting plaintexts.
- **Rogue Ciphertext Security:** It should be computationally infeasible for an adversary to craft a malformed ciphertext that causes decryption of an honest party's ciphertext in the same batch to fail.

Definition 4 (Rogue Ciphertext Security)**.** BTE *is* rogue ciphertext secure *if all non-uniform PPT adversaries $\mathcal{A}$ have a negligible probability of winning the following game:*

- *The challenger runs $(\mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^t)$ and sends everything to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a message $m \in \mathcal{M}$.*
- *The challenger computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$ and sends ct to $\mathcal{A}$.*
- *$\mathcal{A}$ chooses a batch length $B \geq 1$ and outputs ciphertexts $\{\mathsf{ct}_i\}_{i \in [B]}$, a subset $S \subseteq [n]$ with $|S| \geq t$, and partial decryption shares $\{\mathsf{pd}_j\}_{j \in S}$.*
- *The adversary wins the game if:*
  - *there exists $i^* \in [B]$ such that $\mathsf{ct} = \mathsf{ct}_{i^*}$,*
  - *$\mathsf{Verify}(\mathsf{dk}, \ell, \mathsf{pd}_\ell, \{\mathsf{ct}_i\}_{i \in [B]}) = 1$ for all $\ell \in S$,*
  - *$\mathsf{pd} \leftarrow \mathsf{Combine}(\mathsf{dk}, S, \{\mathsf{pd}_j\}_{j \in S})$ succeeds and $(m'_1, \ldots, m'_B) \leftarrow \mathsf{Dec}(\mathsf{dk}, \mathsf{pd}, \{\mathsf{ct}_i\}_{i \in [B]})$,*
  - *$m'_{i^*} = \perp$ or $m'_{i^*} \neq m$.*

Definition 5 (Consistency of BTE)**.** BTE *is* consistent *if all non-uniform PPT adversaries $\mathcal{A}$ have a negligible probability of winning the following game:*

- *The challenger runs $(\mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^t)$ and sends everything to $\mathcal{A}$.*

- *$\mathcal{A}$ chooses a batch length $B \geq 1$ and outputs ciphertexts $\{\mathsf{ct}_i\}_{i\in[B]}$, two subsets $S_1, S_2 \subseteq [n]$ with $|S_1|, |S_2| \geq t$, and partial decryptions $\{\mathsf{pd}_j^1\}_{j\in S_1}$ and $\{\mathsf{pd}_j^2\}_{j\in S_2}$.*
- *The challenger computes $\mathsf{pd}^{(b)} \leftarrow \mathsf{Combine}(\mathsf{dk}, S_b, \{\mathsf{pd}_j^b\}_{j\in S_b})$ for each $b \in \{1,2\}$. If either result is $\perp$, the adversary loses. Otherwise, for each $b \in \{1,2\}$ the challenger computes*

$$(m_1^{(b)}, \ldots, m_B^{(b)}) \leftarrow \mathsf{Dec}(\mathsf{dk}, \mathsf{pd}^{(b)}, \{\mathsf{ct}_i\}_{i\in[B]}).$$

- *The adversary wins if:*
  - $\mathsf{Verify}(\mathsf{dk}, j, \mathsf{pd}_j^b, \{\mathsf{ct}_i\}_{i\in[B]}) = 1$ *for all* $b \in \{1,2\}$ *and all* $j \in S_b$,
  - $m_i^{(1)} \neq m_i^{(2)}$ *for some* $i \in [B]$.

4 Construction

We provide a formal description of our construction in Fig. 1 and discuss how to improve the aggregation complexity below. Done naively, aggregation costs $O(B^2)$ scalar multiplications. We now sketch how it can be computed in $O(B \log^2 B)$ group operations and $O(B)$ pairings. Similar to prior work, the former can be computed while partial decryptions are being collected.

Lemma 1 (Cauchy transform)**.** *Fix a source group $\mathbb{G}$ and group elements $Y_1, \ldots, Y_B \in \mathbb{G}$ and distinct points $x_1, \ldots, x_B \in \mathbb{F}$. Define*

$$F(X) := \prod_{j\in[B]} (X - x_j), \qquad N(X) := \sum_{j\in[B]} Y_j \cdot \frac{F(X)}{X - x_j} \in \mathbb{G}[X].$$

For every $i \in [B]$,

$$N'(x_i) = F'(x_i) \cdot \sum_{j\neq i} \frac{1}{x_i - x_j} \cdot Y_j + \frac{F''(x_i)}{2} \cdot Y_i.$$

Proof. Write $F = (X - x_i)F_i$ with $F_i := \prod_{j\neq i}(X - x_j)$, so that $N = Y_i \cdot F_i + (X - x_i)\sum_{j\neq i} Y_j \cdot \frac{F_i}{X - x_j}$. Differentiating at $X = x_i$,

$$N'(x_i) = F_i'(x_i) \cdot Y_i + \sum_{j\neq i} \frac{F_i(x_i)}{x_i - x_j} \cdot Y_j,$$

and from $F' = F_i + (X - x_i)F_i'$ and $F'' = 2F_i' + (X - x_i)F_i''$ we get $F_i(x_i) = F'(x_i)$ and $F_i'(x_i) = F''(x_i)/2$. □

Now recall that recovering m_i requires computing two MSMs:

$$U_i := \sum_{j\neq i} \frac{1}{x_j - x_i} \cdot \mathsf{ct}_2^{(j)} \in \mathbb{G}_2, \qquad W_i := \sum_{j\neq i} \frac{1}{x_j - x_i} \cdot \mathsf{ct}_3^{(j)} \in \mathbb{G}_1,$$

and a few pairings:

$$m_i = \mathsf{ct}_4^{(i)} - \mathsf{pd} \circ \mathsf{ct}_2^{(i)} + W_i \circ \mathsf{ct}_2^{(i)} - \mathsf{ct}_3^{(i)} \circ U_i. \tag{1}$$

Viewed as linear maps on the vectors $(\mathsf{ct}_2^{(j)})_j$ and $(\mathsf{ct}_3^{(j)})_j$, both MSMs are given by the same matrix C with $C_{ij} = \frac{1}{x_j - x_i}$ for $j \neq i$ and $C_{ii} = 0$. Lemma 1 lets us apply such a transformation in quasi-linear time.

Let $N_2 \in \mathbb{G}_2[X]$ and $N_3 \in \mathbb{G}_1[X]$ be the polynomial N of the lemma for the inputs $Y_j = \mathsf{ct}_2^{(j)}$ and $Y_j = \mathsf{ct}_3^{(j)}$ respectively (the polynomial F is the same for both). Since $\sum_{j\neq i} \frac{Y_j}{x_i - x_j}$ is exactly $-U_i$ (resp. $-W_i$), dividing the lemma by $F'(x_i)$ gives

$$U_i = c_i \cdot \mathsf{ct}_2^{(i)} - \frac{N_2'(x_i)}{F'(x_i)}, \qquad W_i = c_i \cdot \mathsf{ct}_3^{(i)} - \frac{N_3'(x_i)}{F'(x_i)}, \qquad c_i := \frac{F''(x_i)}{2F'(x_i)}.$$

Substituting into Eq. (1), the two c_i terms cancel, leaving

$$m_i = \mathsf{ct}_4^{(i)} - \mathsf{pd} \circ \mathsf{ct}_2^{(i)} - \frac{N_3'(x_i) \circ \mathsf{ct}_2^{(i)} - \mathsf{ct}_3^{(i)} \circ N_2'(x_i)}{F'(x_i)}. \tag{2}$$

For either input vector $(Y_j)_j$, construct F and N using a balanced subproduct tree. At a leaf set $F_{\{j\}} = X - x_j$ and $N_{\{j\}} = Y_j$ and combine disjoint children L, R by

$$F_{L \cup R} = F_L F_R, \qquad N_{L \cup R} = N_L F_R + N_R F_L.$$

Each of the $\log B$ levels of the tree costs $O(B \log B)$ (as we work in an FFT-friendly field), so F, N_2 and N_3 are constructed in $O(B \log^2 B)$ group operations. Differentiation is linear-time, and a remainder tree over the same subproduct tree (reducing F', N_2' and N_3' modulo the scalar polynomials F_S from the root down to the leaves) evaluates them at all x_i within the same bound. Since the points are distinct, all $F'(x_i)$ are nonzero, and batch inversion costs $O(B)$ field operations and one inversion. Thus Eq. (2) gives the claimed $O(B \log^2 B)$ group operations and $O(B)$ pairings.

4.1 Security

Correctness. The derivation in Section 1 shows that Dec recovers every message of a batch of honestly generated ciphertexts whenever the points $x_1, \ldots, x_B$ are pairwise distinct, and Verify accepts the honest shares by bilinearity. It remains to argue that honest points collide only with negligible probability, which is where collision resistance of H is used. The first components $\mathsf{ct}_1^{(i)} = [k_i]_1$ of honest ciphertexts are independent and uniform in $\mathbb{G}_1$, so two of them coincide with probability at most B^2/p. If $H(\mathsf{ct}_1^{(i)}) = H(\mathsf{ct}_1^{(j)})$ for distinct $\mathsf{ct}_1^{(i)} \neq \mathsf{ct}_1^{(j)}$ with noticeable probability, then the algorithm that samples B honest ciphertexts and outputs the colliding pair breaks collision resistance. Hence correctness (Definition 2) holds with overwhelming probability.

Robustness. We turn to robustness. Both properties follow from the pairing check in Verify.

Lemma 2. *Let* $\{\mathsf{ct}^{(i)}\}_{i \in [B]}$ *be a batch and write* $\sum_i \mathsf{ct}_2^{(i)} = [\kappa]_2$. *If* $\mathsf{Verify}(\mathsf{dk}, j, \mathsf{pd}_j, \{\mathsf{ct}^{(i)}\}_{i \in [B]}) = 1$ *then* $\mathsf{pd}_j = [\mathsf{sk}_j \kappa]_1$.

Proof. Verify checks $\mathsf{pd}_j \circ [1]_2 = \mathsf{pk}_j \circ \sum_i \mathsf{ct}_2^{(i)} = [\mathsf{sk}_j \kappa]_T$, and the map $Y \mapsto Y \circ [1]_2$ is injective on $\mathbb{G}_1$ by non-degeneracy. □

In particular, if every ciphertext in the batch is well formed, meaning $\mathsf{ct}_1^{(i)} = [k_i]_1$ and $\mathsf{ct}_2^{(i)} = [k_i]_2$ for some $k_i \in \mathbb{F}$, then $[\kappa]_1 = \sum_i \mathsf{ct}_1^{(i)}$ and the unique accepted response is the honest one, $\mathsf{pd}_j = \mathsf{sk}_j \cdot \sum_i \mathsf{ct}_1^{(i)}$.

Theorem 1 (Consistency)**.** *The construction in Fig. 1 is consistent (Definition 5), unconditionally.*

Proof. Let $\mathcal{A}$ output a batch $\{\mathsf{ct}^{(i)}\}_{i \in [B]}$, two subsets $S_1, S_2 \subseteq [n]$ of size at least t, and shares $\{\mathsf{pd}_j^1\}_{j \in S_1}$ and $\{\mathsf{pd}_j^2\}_{j \in S_2}$ that all verify. By Lemma 2, $\mathsf{pd}_j^b = [\mathsf{sk}_j \kappa]_1$ for every accepted share, where $[\kappa]_2 = \sum_i \mathsf{ct}_2^{(i)}$ depends only on the batch. Interpolating in the exponent, both sets recover the same value,

$$\mathsf{pd}^{(1)} = \sum_{j \in S_1} \lambda_{S_1, j} \cdot [\mathsf{sk}_j \kappa]_1 = [\mathsf{sk}\, \kappa]_1 = \sum_{j \in S_2} \lambda_{S_2, j} \cdot [\mathsf{sk}_j \kappa]_1 = \mathsf{pd}^{(2)}.$$

Since Dec is deterministic, both sets yield the same message vector and $\mathcal{A}$ cannot win. □

Theorem 2 (Rogue ciphertext security)**.** *Assuming* NIZK *is sound, the construction in Fig. 1 is rogue ciphertext secure (Definition 4).*

Batched Threshold Encryption with Transparent Setup

Parameters: $H : \mathbb{G}_1 \to \mathbb{F}$ is a collision-resistant hash function.

$\mathsf{Setup}(1^\lambda, 1^n, 1^t)$:

- Sample the bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \circ) \leftarrow \mathsf{GrpGen}(1^\lambda)$.
- Sample a generator $g' \in \mathbb{G}_1$ with unknown discrete logarithm (e.g., by hashing to the curve).
- Run a DKG to Shamir secret share $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{F}$ among the committee, party j receiving sk_j.
- Output $\mathsf{ek} := \mathsf{pk} := [\mathsf{sk}]_1$ and $\mathsf{dk} := \{\mathsf{pk}_j := [\mathsf{sk}_j]_1\}_{j\in[n]}$.

$\mathsf{Enc}(\mathsf{ek}, m)$: where $m \in \mathbb{G}_T$

- Sample $k \leftarrow\!\!\$\ \mathbb{F}$, set $x := H([k]_1)$, and compute

$$\mathsf{ct} := \left([k]_1,\ [k]_2,\ k \cdot (g' + x \cdot \mathsf{pk}),\ k^2 \cdot [\mathsf{sk}]_T + m\right), \qquad \text{where } [\mathsf{sk}]_T := \mathsf{pk} \circ [1]_2 .$$

- Prove $\pi \leftarrow \left\{k \;\middle|\; \mathsf{ct}_1 = [k]_1 \wedge \mathsf{ct}_2 = [k]_2 \wedge \mathsf{ct}_3 = k \cdot (g' + x \cdot \mathsf{pk}) \wedge \mathsf{ct}_4\right\}$.
- Output (ct, π).

$\mathsf{PartialDec}(\mathsf{ek}, \mathsf{sk}_j, \{\mathsf{ct}^{(i)}\}_{i\in[B]})$: If $\mathsf{NIZK.Verify}(\mathsf{ct}^{(i)}, \pi_i) = 0$ for some $i \in [B]$, or the points $x_1, \ldots, x_B$ are not pairwise distinct, output $\perp$. Otherwise output $\mathsf{pd}_j := \mathsf{sk}_j \cdot \sum_{i=1}^{B} \mathsf{ct}_1^{(i)}$.

$\mathsf{Verify}(\mathsf{dk}, j, \mathsf{pd}_j, \{\mathsf{ct}^{(i)}\}_{i\in[B]})$: If $\mathsf{NIZK.Verify}(\mathsf{ct}^{(i)}, \pi_i) = 0$ for some $i \in [B]$, or the points $x_1, \ldots, x_B$ are not pairwise distinct, output 0. Otherwise output 1 iff $\mathsf{pd}_j \circ [1]_2 = \mathsf{pk}_j \circ \sum_{i=1}^{B} \mathsf{ct}_2^{(i)}$.

$\mathsf{Combine}(\mathsf{dk}, T, \{\mathsf{pd}_j\}_{j\in T})$: If $|T| < t$ output $\perp$. Otherwise interpolate in the exponent and output $\mathsf{pd} := \sum_{j\in T} \lambda_{T,j} \cdot \mathsf{pd}_j \ \left(= \mathsf{sk} \cdot \sum_{i=1}^{B} \mathsf{ct}_1^{(i)}\right)$.

$\mathsf{Dec}(\mathsf{dk}, \mathsf{pd}, \{\mathsf{ct}^{(i)}\}_{i\in[B]})$: For each $i \in [B]$, compute

$$U_i := \sum_{\substack{j\in[B]\\ j\neq i}} \frac{\mathsf{ct}_2^{(j)}}{x_j - x_i} \in \mathbb{G}_2, \qquad W_i := \sum_{\substack{j\in[B]\\ j\neq i}} \frac{\mathsf{ct}_3^{(j)}}{x_j - x_i} \in \mathbb{G}_1,$$

$$m_i := \mathsf{ct}_4^{(i)} - (\mathsf{pd} - W_i) \circ \mathsf{ct}_2^{(i)} - \mathsf{ct}_3^{(i)} \circ U_i.$$

Figure 1: Our BTE scheme with transparent setup proven secure under Definition 6.

Proof. Let $\mathcal{A}$ output a batch $\{\mathsf{ct}^{(i)}\}_{i\in[B]}$ containing the honest ciphertext $\mathsf{ct} = \mathsf{ct}^{(i^*)}$, a set $S \subseteq [n]$ with $|S| \geq t$, and shares $\{\mathsf{pd}_j\}_{j\in S}$ that all verify. Since Verify accepts, every ciphertext proof verifies and the points $x_1, \ldots, x_B$ are pairwise distinct. By soundness of NIZK, except with negligible probability every statement is true. That is, for each $i \in [B]$ there exists $k_i \in \mathbb{F}$ such that

$$\mathsf{ct}_1^{(i)} = [k_i]_1, \qquad \mathsf{ct}_2^{(i)} = [k_i]_2, \qquad \mathsf{ct}_3^{(i)} = k_i \cdot (g' + x_i \cdot \mathsf{pk}),$$

with $x_i = H(\mathsf{ct}_1^{(i)})$. Every ciphertext is therefore well formed, so by Lemma 2 $\mathsf{pd}_j = \mathsf{sk}_j \cdot \sum_i \mathsf{ct}_1^{(i)}$ for all $j \in S$, and Combine outputs $\mathsf{pd} = [\mathsf{sk} \sum_i k_i]_1$. The value m'_{i^*} computed by Dec depends on the remaining ciphertexts only through $\mathsf{ct}_2^{(j)}$ and $\mathsf{ct}_3^{(j)}$ for $j \neq i^*$, which have the prescribed form, and on the pairwise distinct points x_j. The $\mathsf{ct}_4^{(j)}$ components of the other ciphertexts are not used. The derivation in Section 1 therefore applies verbatim and Dec subtracts exactly the mask $[\mathsf{sk}\, k_{i^*}^2]_T$ from $\mathsf{ct}_4^{(i^*)}$, regardless of how the remaining ciphertexts were formed. Since $\mathsf{ct}^{(i^*)} = \mathsf{ct}$ is honest, $\mathsf{ct}_4^{(i^*)} = [\mathsf{sk}\, k_{i^*}^2]_T + m$ and Dec outputs

$m'_{i^*} = m$. Moreover, Dec always outputs elements of $\mathbb{G}_T$ and never $\perp$. Hence $\mathcal{A}$ wins only if soundness of NIZK fails, which happens with negligible probability. □

Chosen-ciphertext security. We prove CCA security under an asymmetric variant of the decisional bilinear square Diffie–Hellman assumption of Heidarvand and Villar [HV09].

Definition 6 (Decisional Bilinear Square Diffie–Hellman (DBSDH))**.** *The* DBSDH *assumption holds for* GrpGen *if the following two distributions are computationally indistinguishable:*

$$\mathcal{D}_0 = \Big([\alpha]_1,\ [k]_1,\ [k]_2,\ \big[\alpha k^2\big]_T\Big),$$
$$\mathcal{D}_1 = \Big([\alpha]_1,\ [k]_1,\ [k]_2,\ [z]_T\Big),$$

where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \circ) \leftarrow \mathsf{GrpGen}(1^\lambda)$ *and* $\alpha, k, z \leftarrow \mathbb{F}$.

Remark 1. *In symmetric groups, Definition 6 coincides with the decisional* 1*-wBDHI* problem of Boneh, Boyen and Goh [BBG05] and the decisional* 1*-BDHE problem of Boneh, Gentry and Waters [BGW05].*

Theorem 3 (B-IND-CCA security)**.** *Assume that the* DBSDH *assumption holds and that* NIZK *is zero-knowledge and straight-line simulation-extractable. Then the construction in Fig. 1 is B-IND-CCA-secure (Definition 3).*

Proof. Let $\mathcal{A}$ be an adversary in the B-IND-CCA game (Definition 3). We write (ct^*, π^*) for the challenge ciphertext, k^* for its randomness and $x^* = H(\mathsf{ct}_1^*)$ for its hash value. Consider the following hybrids.

$\mathbf{H}_1$: This is the real game with challenge bit $b = 0$: the challenge ciphertext is an honest encryption of m_0, i.e.,

$$\mathsf{ct}^* = \big([k^*]_1,\ [k^*]_2,\ k^* \cdot (g' + x^* \cdot \mathsf{pk}),\ (k^*)^2 \cdot [\mathsf{sk}]_T + m_0\big),$$

and π^* is generated honestly.

$\mathbf{H}_2$: Identical to $\mathbf{H}_1$, except that π^* is generated with the simulator of NIZK. Indistinguishable from $\mathbf{H}_1$ by zero-knowledge.

$\mathbf{H}_3$: Identical to $\mathbf{H}_2$, except that $(k^*)^2 \cdot [\mathsf{sk}]_T + m_0$ is replaced by $\mathsf{ct}_4^* := Z + m_0$ for a uniformly random $Z \leftarrow \mathbb{G}_T$.

We claim that $\mathbf{H}_2$ and $\mathbf{H}_3$ are indistinguishable under DBSDH. Let $\mathcal{B}$ receive $([\alpha]_1, [k]_1, [k]_2, Z)$ where $Z = \big[\alpha k^2\big]_T$ or Z is uniform, and let $\mathcal{B}$ simulate the game as follows.

Setup. $\mathcal{B}$ receives the corrupted set S (of size $t-1$) from $\mathcal{A}$ and sets $\mathsf{pk} := [\alpha]_1$, so $\mathsf{sk} = \alpha$ is unknown to $\mathcal{B}$. For each $u \in S$ it samples $\mathsf{sk}_u \leftarrow \mathbb{F}$ and sets $\mathsf{pk}_u := [\mathsf{sk}_u]_1$. Let f be the unique polynomial of degree at most $t-1$ with $f(0) = \alpha$ and $f(u) = \mathsf{sk}_u$ for $u \in S$. $\mathcal{B}$ does not know f, but computes $\mathsf{pk}_j := [f(j)]_1$ for honest $j \notin S$ by Lagrange interpolation in the exponent from $\mathsf{pk} = [f(0)]_1$ and $\{[\mathsf{sk}_u]_1\}_{u \in S}$. Next, $\mathcal{B}$ fixes the first challenge component $\mathsf{ct}_1^* := [k]_1$, computes $x^* := H(\mathsf{ct}_1^*)$, samples $\theta \leftarrow \mathbb{F}$, and sets

$$g' := [\theta]_1 - x^* \cdot \mathsf{pk}.$$

Since θ is uniform, g' is a uniformly random element of $\mathbb{G}_1$, independent of pk, exactly as in the game. $\mathcal{B}$ hands $\mathcal{A}$ the public keys $\mathsf{pk}, \mathsf{dk}, g'$ and the corrupted shares.

Partial decryption queries. On a query $\{\mathsf{ct}^{(i)}\}_{i \in [B]}$, $\mathcal{B}$ applies the rejection rules of the game and of PartialDec: it returns $\perp$ if some proof fails to verify, if two hash values coincide, or, in the post-challenge phase, if the batch contains ct^*. Every remaining ciphertext carries a verifying proof for a statement–proof pair that is not the simulated pair (ct^*, π^*), so by straight-line simulation-extractability $\mathcal{B}$ extracts, except with negligible probability, a witness k_i with $\mathsf{ct}_1^{(i)} = [k_i]_1$ for every $i \in [B]$. It sets $\kappa := \sum_i k_i$, so that $\sum_i \mathsf{ct}_1^{(i)} = [\kappa]_1$, and returns for every honest $j \notin S$

$$\mathsf{pd}_j := \kappa \cdot \mathsf{pk}_j = \big[\kappa\, \mathsf{sk}_j\big]_1 = \mathsf{sk}_j \cdot \sum_i \mathsf{ct}_1^{(i)},$$

which is exactly the honest response, produced without knowledge of sk_j.

Challenge. When $\mathcal{A}$ outputs (m_0, m_1), $\mathcal{B}$ sets

$$\mathsf{ct}^* := \big([k]_1\,,\ [k]_2\,,\ \theta \cdot [k]_1\,,\ Z + m_0\big),$$

simulates π^* for the statement ct^*, adds the pair to the simulator's list, and sends (ct^*, π^*) to $\mathcal{A}$. Since $g' + x^* \cdot \mathsf{pk} = [\theta]_1$, we have $\mathsf{ct}_3^* = [k\theta]_1 = k \cdot (g' + x^* \cdot \mathsf{pk})$, so $(\mathsf{ct}_1^*, \mathsf{ct}_2^*, \mathsf{ct}_3^*)$ is a correctly formed ciphertext with randomness $k^* = k$. If $Z = \big[\alpha k^2\big]_T = \big[\mathsf{sk}\,(k^*)^2\big]_T$ the view of $\mathcal{A}$ is that of $\mathbf{H}_2$ and if Z is uniform it is that of $\mathbf{H}_3$. Finally $\mathcal{B}$ outputs $\mathcal{A}$'s guess. Hence a distinguisher between $\mathbf{H}_2$ and $\mathbf{H}_3$ breaks DBSDH.

Finally, $\mathbf{H}_3$ is message-independent: if Z is uniform in $\mathbb{G}_T$, then $Z + m_0$ and $Z + m_1$ are identically distributed, and the remaining components of ct^* and all oracle answers do not depend on the message. Starting from this hybrid, we apply the same transitions in reverse with m_1 in place of m_0 and arrive at the real game with challenge bit $b = 1$. Therefore the real games for $b = 0$ and $b = 1$ are computationally indistinguishable, and the construction is B-IND-CCA-secure. □

References

[ABD+25] Amit Agarwal, Kushal Babel, Sourav Das, Babak Poorebrahim Gilkalaye, Arup Mondal, Benny Pinkas, Peter Rindal, and Aayush Yadav. Weighted batched threshold encryption with applications to mempool privacy. Cryptology ePrint Archive, Report 2025/2115, 2025. 3

[ACADG26] Amit Agarwal, Rutchathon Chairattana-Apirom, Sourav Das, and Babak Poorebrahim Gilkalaye. Batched and weighted threshold encryption with silent setup. Cryptology ePrint Archive, Report 2026/2087, 2026. 3

[ADG+26] Amit Agarwal, Sourav Das, Babak Poorebrahim Gilkalaye, Peter Rindal, and Victor Shoup. BTX: Simple and efficient batch threshold encryption. Cryptology ePrint Archive, Paper 2026/754, 2026. 3

[AFP25] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 69–100. Springer, Cham, August 2025. 3

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Berlin, Heidelberg, May 2005. 10

[BCF+25] Jan Bormet, Arka Rai Choudhuri, Sebastian Faust, Sanjam Garg, Hussien Othman, Guru-Vamsi Policharla, Ziyan Qu, and Mingyuan Wang. BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. Cryptology ePrint Archive, Report 2025/1419, 2025. 2, 3

[BFOQ25] Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. In Lujo Bauer and Giancarlo Pellegrino, editors, *USENIX Security 2025*, pages 3457–3476. USENIX Association, August 2025. 2, 3

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Berlin, Heidelberg, August 2005. 10

[BLT25] Dan Boneh, Evan Laufer, and Ertem Nusret Tas. Batch decryption without epochs and its application to encrypted mempools. Cryptology ePrint Archive, Report 2025/1254, 2025. 3

[BNRT26] Dan Boneh, Rohit Nema, Arnab Roy, and Ertem Nusret Tas. Efficient batch threshold encryption using partial fraction techniques. Cryptology ePrint Archive, Paper 2026/674, 2026. 3

[CGPP24] Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024. 1, 3

[CGPW25] Arka Rai Choudhuri, Sanjam Garg, Guru-Vamsi Policharla, and Mingyuan Wang. Practical mempool privacy via one-time setup batched threshold encryption. In Lujo Bauer and Giancarlo Pellegrino, editors, *USENIX Security 2025*, pages 3477–3495. USENIX Association, August 2025. 3

[FPRT26] Alexander Frolov, Aditi Partap, Max Resnick, and Ertem Nusret Tas. Weighted batch threshold encryption with efficient DKG. Cryptology ePrint Archive, Report 2026/2053, 2026. 3

[FPTX25] Rex Fernando, Guru-Vamsi Policharla, Andrei Tonkikh, and Zhuolun Xiang. TrX: Encrypted mempools in high performance BFT protocols. Cryptology ePrint Archive, Report 2025/2032, 2025. 3

[GWWW25] Junqing Gong, Brent Waters, Hoeteck Wee, and David J. Wu. Threshold batched identity-based encryption from pairings in the plain model. Cryptology ePrint Archive, Report 2025/2103, 2025. 3

[HV09] Somayeh Heidarvand and Jorge L. Villar. Public verifiability from pairings in secret sharing schemes. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 294–308. Springer, Berlin, Heidelberg, August 2009. 10

[Pol26a] Guru-Vamsi Policharla. Labeled multi-key batched IBE. Cryptology ePrint Archive, Report 2026/1452, 2026. 3

[Pol26b] Guru-Vamsi Policharla. A simple batched threshold encryption scheme. Cryptology ePrint Archive, Paper 2026/760, 2026. 3

[Pol26c] Guru-Vamsi Policharla. Weighted threshold encryption with silent setup. Cryptology ePrint Archive, Report 2026/2148, 2026. 2, 3

[SAS24] Sora Suegami, Shinsaku Ashizawa, and Kyohei Shibano. Constant-cost batched partial decryption in threshold encryption. Cryptology ePrint Archive, Paper 2024/762, 2024. 3

[XZC26] Saisi Xiong, Yijian Zhang, and Jie Chen. Efficient batched IBE from lattices in the standard model. *IACR Communications in Cryptology*, 3(2), 2026. 3