

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.

In the ramp setting, with a gap between the reconstruction threshold t and corruption threshold f as in consensus protocols with $n \geq 3f + 1$, we construct a pairing-free batched threshold encryption scheme from DDH whose partial decryptions are $2B/(t - f)$ group elements, and reduce the aggregation cost of our pairing-based scheme to $O(B \log^2(B/(t - f)))$ 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 $\mathbb{F}$
- in the *ramp* setting, with a gap between the reconstruction threshold t and corruption threshold f as in consensus protocols with $n \geq 3f + 1$ and quorums of $t = 2f + 1$, we obtain:
 - a *pairing-free* BTE scheme from DDH whose partial decryptions are $2B/(t - f)$ group elements, i.e. $O(B/n)$ for $n = 3f + 1$ and $t = 2f + 1$ (Appendix A)
 - a reduction in the aggregation cost of our pairing-based scheme (Fig. 1) from $O(B \log^2 B)$ to $O(B \log^2(B/(t - f)))$ group operations (Appendix B)

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 $sk \leftarrow \mathbb{F}$ and publish $pk = [sk]_1$ and $\{pk_i = [sk_i]_1\}_{i \in [n]}$ 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

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

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

Partial Decryption. To decrypt a batch of ciphertexts

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

the j -th committee member publishes

$$pd_j := sk_j \cdot \sum_{i=1}^B [k_i]_1$$

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

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

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

Thus we can recover the i -th message as:

$$m_i = ct_4^{(i)} - pd \circ [k_i]_2 + \sum_{j \neq i} \frac{(k_j \cdot (g' + x_j \cdot pk)) \circ [k_i]_2 - (k_i \cdot (g' + x_i \cdot pk)) \circ [k_j]_2}{x_j - x_i}$$

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 $pd = (\sum_i k_i)[sk]_1$, and $[sk]_1$ 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

Construction	Setup	$ \text{pp} $	$ \text{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	MPC	$O(Bn)$	$2[\mathbb{G}_1]$	✓	✓	B -StatRankDef
[BNRT26]	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 $|\text{pp}|$ is the total size of the public parameters, including any structured reference string, all public keys, and public aggregation hints. The column $|\text{ct}|$ 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.

¹The 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.

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

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

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.

2 Preliminaries

We use $[n]$ to denote the set $\{1, 2, \dots, n\}$. The security parameter is denoted by $\lambda \in \mathbb{N}$. A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is polynomially bounded if $f(n) = O(n^c)$ for some constant $c \geq 0$ as $n \rightarrow \infty$, and we write $\text{poly}(\cdot)$ to denote a polynomial bound. A function $f : \mathbb{N} \rightarrow [0, 1]$ 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 $\text{negl}(\cdot)$ to denote such a function. A probability is *noticeable* if it is not negligible, and *overwhelming* if it is equal to $1 - \text{negl}(\lambda)$ for some negligible function $\text{negl}(\lambda)$. 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 $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \circ) \leftarrow \text{GrpGen}(1^\lambda)$ 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 $\circ : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ 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:

- $\text{Setup}(1^\lambda) \rightarrow (\text{crs}, \text{td})$: Takes as input a security parameter, and outputs a common reference string crs and trapdoor td .
- $\text{Prove}(\text{crs}, x, w) \rightarrow \pi$: Takes as input crs and any pair $(x, w) \in \mathcal{R}$ and outputs a proof π for the statement $x \in \mathcal{L}$.
- $\text{Verify}(\text{crs}, x, \pi) \rightarrow b$: Takes as input crs , statement x and proof π and outputs a bit b indicating whether verification has passed or failed.
- $\text{SimProve}(\text{crs}, \text{td}, x) \rightarrow \pi$: 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 $\{w \mid \varphi_1 \wedge \dots \wedge \varphi_m\}$ 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

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

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

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

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

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

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

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

where $\mathcal{A}$ has oracle access to $\text{SimProve}(\text{crs}, \text{td}, \cdot)$, 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:

- $\text{Setup}(1^\lambda, 1^n, 1^t) \rightarrow (\text{ek}, \text{sk}_1, \dots, \text{sk}_n, \text{dk})$: 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 $(\text{sk}_1, \dots, \text{sk}_n)$.
- $\text{Enc}(\text{ek}, m) \rightarrow \text{ct}$: The encryption algorithm takes the encryption key ek and a message $m \in \mathcal{M}$, and outputs a ciphertext ct .
- $\text{PartialDec}(\text{ek}, \text{sk}_j, \{\text{ct}_i\}_{i \in [B]}) \rightarrow \text{pd}_j$: 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 $\perp$.
- $\text{Verify}(\text{dk}, j, \text{pd}_j, \{\text{ct}_i\}_{i \in [B]}) \rightarrow \{0, 1\}$: 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.
- $\text{Combine}(\text{dk}, T \subseteq [n], \{\text{pd}_j\}_{j \in T}) \rightarrow \text{pd}$: 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 $\perp$.
- $\text{Dec}(\text{dk}, \text{pd}, \{\text{ct}_i\}_{i \in [B]}) \rightarrow \{m_i\}_{i \in [B]}$: 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 $\text{BTE} = (\text{Setup}, \text{Enc}, \text{PartialDec}, \text{Verify}, \text{Combine}, \text{Dec})$ 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 $\{m_i\}_{i \in [B]} \in \mathcal{M}^B$,

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

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 $(\text{ek}, \text{sk}_1, \dots, \text{sk}_n, \text{dk}) \leftarrow \text{Setup}(1^\lambda, 1^n, 1^t)$ and sends ek , dk , and $\{\text{sk}_j\}_{j \in S}$ 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 $\{\text{ct}_i\}_{i \in [B]}$ of B ciphertexts. The challenger computes $\text{pd}_j := \text{PartialDec}(\text{ek}, \text{sk}_j, \{\text{ct}_i\}_{i \in [B]})$ 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 $\text{ct}^* \leftarrow \text{Enc}(\text{ek}, m_b)$, 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 $\{\text{ct}_i\}_{i \in [B]}$ such that $\text{ct}^* \in \{\text{ct}_i\}_{i \in [B]}$, 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

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

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 $(\text{ek}, \text{sk}_1, \dots, \text{sk}_n, \text{dk}) \leftarrow \text{Setup}(1^\lambda, 1^n, 1^t)$ and sends everything to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message $m \in \mathcal{M}$.
- The challenger computes $\text{ct} \leftarrow \text{Enc}(\text{ek}, m)$ and sends ct to $\mathcal{A}$.
- $\mathcal{A}$ chooses a batch length $B \geq 1$ and outputs ciphertexts $\{\text{ct}_i\}_{i \in [B]}$, a subset $S \subseteq [n]$ with $|S| \geq t$, and partial decryption shares $\{\text{pd}_j\}_{j \in S}$.
- The adversary wins the game if:
 - there exists $i^* \in [B]$ such that $\text{ct} = \text{ct}_{i^*}$,

- $\text{Verify}(\text{dk}, \ell, \text{pd}_\ell, \{\text{ct}_i\}_{i \in [B]}) = 1$ for all $\ell \in S$,
- $\text{pd} \leftarrow \text{Combine}(\text{dk}, S, \{\text{pd}_j\}_{j \in S})$ succeeds and $(m'_1, \dots, m'_B) \leftarrow \text{Dec}(\text{dk}, \text{pd}, \{\text{ct}_i\}_{i \in [B]})$,
- $m'_{i^*} = \perp$ 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 $(\text{ek}, \text{sk}_1, \dots, \text{sk}_n, \text{dk}) \leftarrow \text{Setup}(1^\lambda, 1^n, 1^t)$ and sends everything to $\mathcal{A}$.
- $\mathcal{A}$ chooses a batch length $B \geq 1$ and outputs ciphertexts $\{\text{ct}_i\}_{i \in [B]}$, two subsets $S_1, S_2 \subseteq [n]$ with $|S_1|, |S_2| \geq t$, and partial decryptions $\{\text{pd}_j^1\}_{j \in S_1}$ and $\{\text{pd}_j^2\}_{j \in S_2}$.
- The challenger computes $\text{pd}^{(b)} \leftarrow \text{Combine}(\text{dk}, S_b, \{\text{pd}_j^b\}_{j \in S_b})$ for each $b \in \{1, 2\}$. If either result is $\perp$, the adversary loses. Otherwise, for each $b \in \{1, 2\}$ the challenger computes

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

- The adversary wins if:
 - $\text{Verify}(\text{dk}, j, \text{pd}_j^b, \{\text{ct}_i\}_{i \in [B]}) = 1$ 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 $Y_1, \dots, Y_B \in \mathbb{G}$ and distinct points $x_1, \dots, x_B \in \mathbb{F}$. Define

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

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 $F'_i(x_i) = F''(x_i)/2$. $\square$

Now recall that recovering m_i requires computing two MSMs:

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

and a few pairings:

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

Viewed as linear maps on the vectors $(\text{ct}_2^{(j)})_j$ and $(\text{ct}_3^{(j)})_j$, 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 $Y_j = \text{ct}_2^{(j)}$ and $Y_j = \text{ct}_3^{(j)}$ 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

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

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

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

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

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

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. In the ramp setting, the batch can be split into B/ℓ sub-batches of $\ell = \lfloor (t - f + 1)/2 \rfloor$ ciphertexts, and the transform runs independently on each of the ℓ slots of B/ℓ ciphertexts, see Appendix B.

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 $x_1, \dots, x_B$ 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 $\text{ct}_1^{(i)} = [k_i]_1$ of honest ciphertexts are independent and uniform in $\mathbb{G}_1$, so two of them coincide with probability at most B^2/p . If $H(\text{ct}_1^{(i)}) = H(\text{ct}_1^{(j)})$ for distinct $\text{ct}_1^{(i)} \neq \text{ct}_1^{(j)}$ 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 $\{\text{ct}^{(i)}\}_{i \in [B]}$ be a batch and write $\sum_i \text{ct}_2^{(i)} = [\kappa]_2$. If $\text{Verify}(\text{dk}, j, \text{pd}_j, \{\text{ct}^{(i)}\}_{i \in [B]}) = 1$ then $\text{pd}_j = [\text{sk}_j \kappa]_1$.*

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

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

Batched Threshold Encryption with Transparent Setup

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

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

- Sample the bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \circ) \leftarrow \text{GrpGen}(1^\lambda)$.
- 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 $\text{sk} \leftarrow \mathbb{F}$ among the committee, party j receiving sk_j .
- Output $\text{ek} := \text{pk} := [\text{sk}]_1$ and $\text{dk} := \{\text{pk}_j := [\text{sk}_j]_1\}_{j \in [n]}$.

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

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

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

- Prove $\pi \leftarrow \{k \mid \text{ct}_1 = [k]_1 \wedge \text{ct}_2 = [k]_2 \wedge \text{ct}_3 = k \cdot (g' + x \cdot \text{pk}) \wedge \text{ct}_4\}$.
- Output (ct, π) .

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

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

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

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

$$U_i := \sum_{\substack{j \in [B] \\ j \neq i}} \frac{\text{ct}_2^{(j)}}{x_j - x_i} \in \mathbb{G}_2, \quad W_i := \sum_{\substack{j \in [B] \\ j \neq i}} \frac{\text{ct}_3^{(j)}}{x_j - x_i} \in \mathbb{G}_1, \\ m_i := \text{ct}_4^{(i)} - (\text{pd} - W_i) \circ \text{ct}_2^{(i)} - \text{ct}_3^{(i)} \circ U_i.$$

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

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

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

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

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

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

Proof. Let $\mathcal{A}$ output a batch $\{\text{ct}^{(i)}\}_{i \in [B]}$ containing the honest ciphertext $\text{ct} = \text{ct}^{(i^*)}$, a set $S \subseteq [n]$ with $|S| \geq t$, and shares $\{\text{pd}_j\}_{j \in S}$ that all verify. Since Verify accepts, every ciphertext proof verifies and the points

$x_1, \dots, x_B$ 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

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

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

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:*

$$\begin{aligned} \mathcal{D}_0 &= ([\alpha]_1, [k]_1, [k]_2, [\alpha k^2]_T), \\ \mathcal{D}_1 &= ([\alpha]_1, [k]_1, [k]_2, [z]_T), \end{aligned}$$

where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \circ) \leftarrow \text{GrpGen}(1^\lambda)$ 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 $x^* = H(\text{ct}_1^*)$ for its hash value. Consider the following hybrids.

H₁: This is the real game with challenge bit $b = 0$: the challenge ciphertext is an honest encryption of m_0 , i.e.,

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

and π^* is generated honestly.

H₂: Identical to **H₁**, except that π^* is generated with the simulator of NIZK. Indistinguishable from **H₁** by zero-knowledge.

H₃: Identical to **H₂**, except that $(k^*)^2 \cdot [\text{sk}]_T + m_0$ is replaced by $\text{ct}_4^* := Z + m_0$ for a uniformly random $Z \leftarrow \mathbb{G}_T$.

We claim that **H₂** and **H₃** are indistinguishable under DBSDH. Let $\mathcal{B}$ receive $([\alpha]_1, [k]_1, [k]_2, Z)$ where $Z = [\alpha k^2]_T$ 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 $\text{pk} := [\alpha]_1$, so $\text{sk} = \alpha$ is unknown to $\mathcal{B}$. For each $u \in S$ it samples $\text{sk}_u \leftarrow \mathbb{F}$ and sets $\text{pk}_u := [\text{sk}_u]_1$. Let f be the unique polynomial of degree at most $t - 1$ with $f(0) = \alpha$ and $f(u) = \text{sk}_u$ for $u \in S$. $\mathcal{B}$ does not know f , but computes $\text{pk}_j := [f(j)]_1$

for honest $j \notin S$ by Lagrange interpolation in the exponent from $\text{pk} = [f(0)]_1$ and $\{\text{sk}_u\}_u \in S$. Next, $\mathcal{B}$ fixes the first challenge component $\text{ct}_1^* := [k]_1$, computes $x^* := H(\text{ct}_1^*)$, samples $\theta \leftarrow \mathbb{F}$, and sets

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

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 pk, dk, g' and the corrupted shares.

Partial decryption queries. On a query $\{\text{ct}^{(i)}\}_{i \in [B]}$, $\mathcal{B}$ 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 $\text{ct}_1^{(i)} = [k_i]_1$ for every $i \in [B]$. It sets $\kappa := \sum_i k_i$, so that $\sum_i \text{ct}_1^{(i)} = [\kappa]_1$, and returns for every honest $j \notin S$

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

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

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

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

simulates π^* for the statement ct^* , adds the pair to the simulator’s list, and sends (ct^*, π^*) to $\mathcal{A}$. Since $g' + x^* \cdot \text{pk} = [\theta]_1$, we have $\text{ct}_3^* = [k\theta]_1 = k \cdot (g' + x^* \cdot \text{pk})$, so $(\text{ct}_1^*, \text{ct}_2^*, \text{ct}_3^*)$ is a correctly formed ciphertext with randomness $k^* = k$. If $Z = [\alpha k^2]_T = [\text{sk}(k^*)^2]_T$ 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. $\square$

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](#), [4](#)
- [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](#), [4](#)
- [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](#), [16](#)
- [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, Ziyang 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 Ziyang 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 Wenyan 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](#)
- [DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 445–465. Springer, Berlin, Heidelberg, May / June 2010. [13](#)
- [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](#), [4](#)
- [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](#)
- [FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *24th ACM STOC*, pages 699–710. ACM Press, May 1992. [13](#)
- [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](#), [16](#)
- [Pol26c] Guru-Vamsi Policharla. Weighted threshold encryption with silent setup. Cryptology ePrint Archive, Report 2026/2148, 2026. [3](#), [4](#)

- [SAS24] Sora Suegami, Shinsaku Ashizawa, and Kyohei Shibano. Constant-cost batched partial decryption in threshold encryption. *Cryptology ePrint Archive*, Paper 2024/762, 2024. [3](#)
- [Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. [13](#)
- [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](#)

A Pairing-Free Batched Threshold Encryption

In the ramp setting with a gap between the corruption threshold f and reconstruction threshold t , we construct a pairing-free batched threshold encryption scheme, secure under DDH.

Packed secret sharing. Franklin and Yung [FY92] observed that with a gap between the two thresholds, secrets can be shared more efficiently than with Shamir secret sharing [Sha79]. Concretely, one can pack ℓ secrets into n shares, with reconstruction from any $f + \ell$ of them, as follows:

- Choose two non-intersecting domains, D of size ℓ and D_s of size n . We use $D = [\ell] = \{1, \dots, \ell\}$ and $D_s = [n^-] = \{-1, \dots, -n\}$.
- Sample a random polynomial $S(X)$ of degree at most $f + \ell - 1$ which evaluates to $(s_1, \dots, s_\ell)$ on D .
- The *packed* secret sharing is $\{S(-i)\}_{i \in [n]}$.

Any f shares are independent of the secrets, since S has f degrees of freedom left once its values on D are fixed, and any $f + \ell$ shares determine S . It is linearly homomorphic, and multiplying two sharings pointwise gives a sharing of the Hadamard product of the secrets. This has been used extensively in secure multiparty computation [FY92, DIK10] to reduce asymptotic communication and computation complexity. In hindsight, it's not surprising that these techniques extend to BTE as decryption is just a special-purpose multi-party computation protocol.

We now construct a pairing-free batched threshold encryption scheme over a group $\mathbb{G} = \langle g \rangle$ of prime order p with message space $\mathbb{G}$. For a quorum size $t \leq n - f$, we can support any corruption threshold f and batch size ℓ satisfying:

$$t \geq f + 2\ell - 1$$

- **Setup:** Sample $\text{sk} \leftarrow \$ \mathbb{F}$ and a random polynomial $S(X)$ of degree at most $f + \ell - 1$ satisfying $S(j) = \text{sk}$ for all $j \in [\ell]$. Party i receives the share $\text{sk}_i = S(-i)$. Publish $\text{pp} = (\text{pk}, \{\text{vk}_i\}_{i \in [n]})$ with $\text{pk} = g^{\text{sk}}$ and $\text{vk}_i = g^{\text{sk}_i}$.
- **Encrypt:** Sample $r \leftarrow \$ \mathbb{F}$, set $c_1 = g^r$ and $c_2 = \text{pk}^r \cdot M$, and output $\text{ct} = (c_1, c_2, \pi)$ where π is a zero-knowledge, straight-line simulation-extractable proof of knowledge for

$$\pi = \{ r \mid c_1 = g^r \wedge c_2 \}$$

- **PartDec:** To decrypt a batch of ciphertexts $\{\text{ct}_j = (c_{1,j}, c_{2,j}, \pi_j)\}_{j \in [\ell]}$, where $c_{1,j} = g^{r_j}$ and $c_{2,j} = \text{pk}^{r_j} \cdot M_j$, first verify every π_j . A ciphertext whose proof fails is replaced by the fixed dummy $(c_{1,j}, c_{2,j}) = (1_{\mathbb{G}}, 1_{\mathbb{G}})$. Then:
 - Let $R(X)$ be the polynomial of degree less than ℓ with $R(j) = r_j$ for $j \in [\ell]$. Party i computes $g^{r'_i}$ for $r'_i = R(-i)$ by Lagrange interpolation in the exponent, using an MSM of ℓ terms with public coefficients.

- The i -th party publishes $\text{pd}_i = (d_i, \pi'_i)$ with $d_i = g^{\text{sk}_i r'_i}$, where π'_i is a sigma-protocol proof for

$$\pi'_i = \{ \text{sk}_i \mid d_i = (g^{r'_i})^{\text{sk}_i} \wedge \text{vk}_i = g^{\text{sk}_i} \}.$$

- **PartVerify:** Verify the proof π'_i against $g^{r'_i}$ and vk_i .
- **Combine:** Given a quorum of t verified partial decryptions, interpolate to recover the polynomial $S(X) \cdot R(X)$ in the exponent and then evaluate on the domain $[\ell]$ to output

$$\text{pd} = (\text{pd}^{(1)}, \dots, \text{pd}^{(\ell)}) = (g^{\text{sk} \cdot r_1}, \dots, g^{\text{sk} \cdot r_\ell}).$$

Note that this can be computed in $O(n \log n)$ time via FFTs when domains are chosen to be roots of unity.

- **Dec:** Given pd and the batch, output $M_j = c_{2,j}/\text{pd}^{(j)}$ for every $j \in [\ell]$ whose proof verified, and $\perp$ for the others.

When the number of transactions $B > \ell$, they can be partitioned and decrypted as separate batches of ℓ each, but the partial decryptions grow to $\lceil B/\ell \rceil$ group elements, which results in a total communication of $O(B)$ for $\ell = \Theta(n)$. Since the same sk_i is used across all sub-batches, a single proof of two $\mathbb{F}$ elements suffices irrespective of the batch size B . With domains chosen as roots of unity, Combine + Dec then costs $O(B \log n)$ group operations for $\ell = \Theta(n)$, and the same bound holds for verifying all partial decryptions.

Security. The definitions of Section 3 carry over with one change: the adversary in Definition 3 corrupts a set of at most f parties rather than exactly $t - 1$, while Combine continues to require t verified partial decryptions.

Theorem 4. *Let $t \geq f + 2\ell - 1$. If π'_i and π are zero-knowledge and straight-line simulation-extractable, the scheme above is correct, consistent, rogue ciphertext secure, and B-IND-CCA secure against f corruptions under DDH in $\mathbb{G}$.*

Proof sketch. An honest partial decryption is $g^{S(-i)R(-i)}$, a point on the polynomial $S \cdot R$ of degree at most $(f + \ell - 1) + (\ell - 1) = f + 2\ell - 2 < t$. Any t such points determine it, and $(S \cdot R)(j) = \text{sk} \cdot r_j$ for $j \in [\ell]$ since S is constant on $[\ell]$.

Consistency. By soundness of π'_i , every partial decryption that verifies has the form $g^{\text{sk}_i r'_i} = g^{S(-i)R(-i)}$, where R is determined by the batch alone. Two quorums therefore interpolate the same polynomial $S \cdot R$ and output the same values on $[\ell]$.

Rogue ciphertexts. Whatever the other ciphertexts are, even dummies, R is defined by $R(j) = r_j$ where $c_{1,j} = g^{r_j}$, so the value recovered at an honest position j is $g^{\text{sk} \cdot r_j}$ and $c_{2,j}/g^{\text{sk} \cdot r_j} = M_j$. Malformed neighbours cannot prevent an honest ciphertext from being decrypted.

B-IND-CCA. Follow the hybrids of Theorem 3: first simulate the challenge proof π^* , then replace the mask. For the second step, given a DDH challenge (g, g^a, g^b, Z) with $Z = g^{ab}$ or uniform, the reduction $\mathcal{B}$ sets $\text{pk} = g^a$. For the corrupt set C with $|C| \leq f$, it samples $\text{sk}_i \leftarrow \mathbb{F}$ for $i \in C$ and uniform values for S at $f - |C|$ further points of D_s ; together with $S(j) = a$ on $[\ell]$ these are $f + \ell$ constraints on a polynomial of degree at most $f + \ell - 1$, so they determine S , and they are distributed exactly as in the real sharing since any f shares of a packed sharing are uniform. $\mathcal{B}$ hands $\mathcal{A}$ the shares sk_i for $i \in C$ and computes every $\text{vk}_i = g^{S(-i)}$ by interpolation in the exponent from g^a (at the points of $[\ell]$) and the f chosen values.

The challenge is $(g^b, Z \cdot M_\beta)$ with a simulated π^* , where β is the challenge bit. It is a real encryption of M_β with randomness $r^* = b$ when $Z = g^{ab}$ and independent of β otherwise. On a partial-decryption query, $\mathcal{B}$ extracts r_j from every ciphertext whose proof verifies, which straight-line simulation-extractability guarantees for every statement-proof pair other than the simulated (ct^*, π^*) , sets $r_j = 0$ for the dummies, computes $R(-i)$ in the field, and answers $\text{vk}_i^{R(-i)} = g^{\text{sk}_i r'_i}$ for every honest party i without knowing sk_i , together with a simulated π'_i . Batches containing ct^* are rejected by the game, so the challenge never has to be answered. $\square$

B Faster Decryption in the Ramp Setting

In Fig. 1 Combine reconstructs a single value $\text{pd} = \text{sk} \cdot \sum_{i \in [B]} \text{ct}_1^{(i)}$ and the cost of aggregation is dominated by the Cauchy transform over all B ciphertexts which requires $O(B \log^2 B)$ operations. With packed secret sharing, the batch of B ciphertexts can be divided into B/ℓ sub-batches of size ℓ (we assume ℓ divides B for simplicity), writing $\text{ct}^{(s,u)}$ for the ciphertext in slot s of the u -th sub-batch, and the partial decryptions (one $\mathbb{G}_1$ element per party) can be used to reconstruct ℓ values

$$\left\{ \text{pd}^{(s)} = \text{sk} \cdot \sum_u \text{ct}_1^{(s,u)} \right\}_{s \in [\ell]},$$

which can be viewed as a reconstructed partial decryption for each of the ℓ slots. Cross terms only arise between ciphertexts in the same slot, so the Cauchy transform runs independently on each slot, resulting in a Combine + Dec complexity of $O(n \log n + B \log^2(B/\ell))$. Ciphertexts, encryption, public parameters and the size of partial decryptions are unchanged. Verifying partial decryptions now requires evaluating R_2 at every party's point. With the domains $[\ell]$ and $\{-1, \dots, -n\}$ replaced by disjoint cosets of roots of unity, this is an inverse FFT of size ℓ followed by an FFT of size n , so verifying all partial decryptions costs $O(n \log n)$ group operations and $2t$ pairings. As in the pairing-free construction, we need $t \geq f + 2\ell - 1$.

- **Setup**($1^\lambda, 1^n, 1^f, 1^t$): Run a DKG to obtain a packed sharing of $\text{sk} \leftarrow \mathbb{F}$: a random polynomial $S(X)$ of degree at most $f + \ell - 1$ with $S(s) = \text{sk}$ for all $s \in [\ell]$, party j receiving $\text{sk}_j = S(-j)$. Output $\text{ek} = \text{pk} = [\text{sk}]_1$ and $\text{dk} = \{\text{pk}_j = [\text{sk}_j]_1\}_{j \in [n]}$.

- **Enc**(ek, m):

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

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

- Prove $\pi \leftarrow \{k \mid \text{ct}_1 = [k]_1 \wedge \text{ct}_2 = [k]_2 \wedge \text{ct}_3 = k \cdot (g' + x \cdot \text{pk}) \wedge \text{ct}_4\}$.
- Output (ct, π) .

- **PartialDec**($\text{ek}, \text{sk}_j, \{\text{ct}^{(s,u)}\}$): Output $\perp$ if some ciphertext proof fails to verify or the points $x_{s,u} = H(\text{ct}_1^{(s,u)})$ are not pairwise distinct. Otherwise let $R_1(X) \in \mathbb{G}_1[X]$ be the polynomial of degree less than ℓ with

$$R_1(s) = \sum_u \text{ct}_1^{(s,u)} \quad \text{for } s \in [\ell].$$

Party j computes its own evaluation $R_1(-j)$ by Lagrange interpolation, an MSM of ℓ terms with public coefficients, and outputs

$$\text{pd}_j = \text{sk}_j \cdot R_1(-j).$$

- **Verify**($\text{dk}, j, \text{pd}_j, \{\text{ct}^{(s,u)}\}$): Perform analogous checks on the batch. Let $R_2(X) \in \mathbb{G}_2[X]$ be the polynomial of degree less than ℓ with $R_2(s) = \sum_u \text{ct}_2^{(s,u)}$ for $s \in [\ell]$, and output 1 iff

$$\text{pd}_j \circ [1]_2 = \text{pk}_j \circ R_2(-j).$$

- **Combine**($\text{dk}, T, \{\text{pd}_j\}_{j \in T}$): Output $\perp$ if $|T| < t$. Otherwise interpolate the polynomial $S(X) \cdot R_1(X)$ in the exponent from the points $\{(-j, \text{pd}_j)\}_{j \in T}$, evaluate it on $[\ell]$ to obtain

$$\text{pd}^{(s)} = \text{sk} \cdot \sum_u \text{ct}_1^{(s,u)} \quad \text{for } s \in [\ell],$$

and output $\text{pd} = (\text{pd}^{(1)}, \dots, \text{pd}^{(\ell)})$.

- **Dec**(dk, pd, $\{\text{ct}^{(s,u)}\}$): For each slot $s \in [\ell]$ and each $u \in [B/\ell]$, compute

$$U_{s,u} = \sum_{u' \neq u} \frac{\text{ct}_2^{(s,u')}}{x_{s,u'} - x_{s,u}} \in \mathbb{G}_2, \quad W_{s,u} = \sum_{u' \neq u} \frac{\text{ct}_3^{(s,u')}}{x_{s,u'} - x_{s,u}} \in \mathbb{G}_1,$$

$$m_{s,u} = \text{ct}_4^{(s,u)} - (\text{pd}^{(s)} - W_{s,u}) \circ \text{ct}_2^{(s,u)} - \text{ct}_3^{(s,u)} \circ U_{s,u}.$$

With $n = 3f + 1$ and $t = 2f + 1$ the packing factor is $\ell = \lfloor (f + 2)/2 \rfloor \approx n/6$, and with $n = 5f + 1$ and $t = 4f + 1$ it is $\ell \approx 3n/10$. The proofs of Section 4.1 go through with the following changes, with the adversary now corrupting at most f parties. For consistency, write $R_2(-j) = [\kappa_j]_2$; as in Lemma 2 an accepted pd_j equals $[\text{sk}_j \kappa_j]_1$, and $\kappa_j = \bar{R}(-j)$ for the polynomial $\bar{R} \in \mathbb{F}[X]$ of degree less than ℓ determined by the batch. Every accepted share therefore lies on $S \cdot \bar{R}$, of degree less than t , so any two quorums recover the same $\text{pd}^{(s)} = [\text{sk} \bar{R}(s)]_1$. For rogue ciphertexts, soundness of NIZK makes the batch well formed, so $\text{pd}^{(s)} = \text{sk} \cdot \sum_u \text{ct}_1^{(s,u)}$, and the honest message is recovered from $\text{pd}^{(s)}$ and the ciphertexts in its own slot exactly as in Theorem 2.

For B-IND-CCA, the reduction of Theorem 3 is modified in two places. Given the corrupt set C with $|C| \leq f$, it samples $\text{sk}_j \leftarrow \mathbb{F}$ for $j \in C$ and uniform values of S at $f - |C|$ more points of $\{-1, \dots, -n\}$. Together with $S(s) = \alpha$ on $[\ell]$ this determines S , with the same distribution as a real packed sharing, and every $\text{pk}_j = [S(-j)]_1$ is computed by interpolation in the exponent from $\text{pk} = [\alpha]_1$ and the f chosen values. On a partial-decryption query, it extracts $k_{s,u}$ for every ciphertext as before, computes κ_j with $R_1(-j) = [\kappa_j]_1$ by interpolating in the field from the slot sums $\sum_u k_{s,u}$, and answers $\text{pd}_j = \kappa_j \cdot \text{pk}_j$ for every honest j . The challenge and the hybrid argument are unchanged.

Remark 2. The same idea reduces the setup size and decryption complexity of Simple BTE [Pol26b] and BTX [ADG⁺26], where the parties hold sharings of the correlated powers $\tau, \dots, \tau^B$ and the public parameters have size $O(Bn)$. Splitting the batch into B/ℓ sub-batches of size ℓ , it suffices to hold a packed sharing of each of the powers $\tau, \dots, \tau^{B/\ell}$, where the u -th power is applied to every ciphertext of the u -th sub-batch. A partial decryption is again a single group element, from which the Simple BTE aggregate of each slot is reconstructed as above, and each slot is then opened as in Simple BTE. This shrinks the MPC setup, the secret state of each party and the public parameters by a factor of ℓ , reduces the cost of verifying a partial decryption from $B + 1$ to $B/\ell + 1$ pairings and that of opening the batch from $O(B \log B)$ to $O(B \log(B/\ell))$ group operations.